Removing Erasures with Explainable Hash Proof Systems

Michel Abdalla, Fabrice Benhamouda, and David Pointcheval

ENS, Paris, France *

Abstract. An important problem in secure multi-party computation is the design of protocols that can tolerate adversaries that are capable of corrupting parties dynamically and learning their internal states. In this paper, we make significant progress in this area in the context of password-authenticated key exchange (PAKE) and oblivious transfer (OT) protocols. More precisely, we first revisit the notion of projective hash proofs and introduce a new feature that allows us to *explain* any message sent by the simulator in case of corruption, hence the notion of *Explainable Projective Hashing*. Next, we demonstrate that this new tool generically leads to efficient PAKE and OT protocols that are secure against semi-adaptive adversaries without erasures in the Universal Composability (UC) framework. We then show how to make these protocols secure even against adaptive adversaries, using *non-committing encryption*, in a much more efficient way than generic conversions from semi-adaptive to adaptive security. Finally, we provide concrete instantiations of explainable projective hash functions that lead to the most efficient PAKE and OT protocols known so far, with UC-security against adaptive adversaries, with or without erasures, in the single global CRS setting.

Keywords. Oblivious Transfer, Erasures, Universal Composability, Adaptive Adversaries.

1 Introduction

1.1 Motivation

One of the most difficult problems in secure multi-party computation is the design of protocols that can tolerate adaptive adversaries. These are adversaries which can corrupt parties dynamically and learn their internal states. As stated in the seminal work of Canetti *et al.* [CFGN96], this problem is even more difficult when uncorrupted parties may deviate from the protocol and keep record of past configurations, instead of erasing them. To deal with this problem, they introduced the concept of non-committing encryption (NCE) and showed how to use it to build general multi-party computation protocols that remained secure even in the presence of such adversaries. Unfortunately, the gain in security came at the cost of a significant loss in efficiency. Though these results were later improved (e.g, [Bea97b,DN00]), obtaining efficient constructions with adaptive security without assuming reliable erasures remains a difficult task.

To address the efficiency issue with previous solutions, Garay *et al.* [GWZ09] introduced two new notions. The first one was the notion of semi-adaptive security in which an adversary is not allowed to corrupt a party if all the parties are honest at the beginning of the protocol. The main advantage of the new notion is that it is only slightly more difficult to achieve than static security but significantly easier than fully adaptive security. The second new notion was the the concept *somewhat non-committing encryption*. Unlike standard NCE schemes, somewhat non-committing encryption only allows the sender of a ciphertext to open it in a limited number of ways, according to an equivocality parameter ℓ.

In addition to being able to build very efficient somewhat non-committing encryption schemes for small values of ℓ, Garay *et al.* [GWZ09] also showed how to build a generic compiler with the help of such schemes that converts any semi-adaptively secure cryptographic scheme into a fully adaptively secure one. Since the equivocality parameter ℓ needed by their compiler is proportional to the input and output domains of the functionality being achieved, they were able to obtain very efficient constructions for functionalities with small domains, such as 1-out-of-2 oblivious transfers (OT). In particular, their results do not assume reliable erasures and hold in the universal composability (UC) framework [Can01, Can00].

Building on the results of Garay *et al.* [GWZ09], Canetti *et al.* [CDVW12] showed how to use 1-out-of-2 OT protocols to build reasonably efficient password-based authenticated key exchange

* DI/ENS, CNRS, and INRIA

(PAKE) protocols in the UC framework against adaptive corruptions without erasures. The number of OT instances used in their protocol is proportional to the number of bits of the password.

Even though both works provide efficient constructions of UC-secure OT and PAKE schemes with adaptive security without erasures, the efficiency gap between these protocols and those which assume reliable erasures (e.g., [CKWZ13, ABB+13]) remains significant. In this work, we aim to reduce this gap.

1.2 Our Approach

In order to build more efficient OT and PAKE schemes with adaptive security without erasures, we first revisit the OT and PAKE constructions of [ABB+13] and show how to make them UC-secure against *semi-adaptive* adversaries, without erasures. Towards this goal, we make two important modifications.

First, we develop a new feature for smooth projective hash functions (SPHFs) [CS02, GL03], that we call *explainability*. Informally speaking, an SPHF is explainable if there exists a trapdoor which allows us to recover a hashing key hk that is compatible with the view of the adversary. Second, we modify the underlying commitment scheme so that the latter remains adaptively secure even without assuming reliable erasures. As we will see later, these two ingredients are sufficient to obtain OT and PAKE constructions that are UC-secure against *semi-adaptive* adversaries.

Finally, we show how to efficiently enhance these protocols with *non-committing encryption* (NCE) in order to achieve adaptive security without erasures. Here, we remark that our use of NCE for migrating from semi-adaptive to adaptive security is specific to our protocols and is more efficient that the generic conversion of Garay *et al.* [GWZ09]. In particular, it makes use of a quasi-optimal number of bits sent via the NCE.

1.3 Basic Tools

The general technique from [ABB+13] was, as in [CHK+05], to use an extractable commitment and a SPHF on the language of valid commitments on a specific word (the password for PAKE and the index-query for OT). However, in order to achieve UC-security in one round, the commitment must also be equivocable. In this case, the language may be trivial since equivocability means that a commitment can be opened in any way. Specific properties on such equivocable and extractable ($\mathtt{E}^2$) commitments have been defined, and namely a robustness property makes them compatible with SPHFs, hence the name of SPHF-friendly commitments.

Equivocable and Extractable Commitments. A commitment is a classical primitive that allows a sender to commit on a value x in such a way that the receiver has no idea about the committed value x (the *hiding* property), whereas the sender can only later open on x (the *binding* property). Equivocability means that a trapdoor allows to open a commitment in any way, and thus implies the perfectly hiding property. While extractability means that a trapdoor allows to extract the committed value from any commitment, which thus implies the perfectly binding property.

The main tool for our purpose is a non-interactive commitment scheme that is simultaneously *equivocable* and *extractable*, but still *hiding* and *binding* even with access to the above oracles for equivocability and extractability. The first candidate is the Canetti and Fischlin [CF01] construction that is unfortunately linear in the bit-length of the committed message, since the bits are essentially individually committed by a perfectly hiding (and equivocable) commitment, while the associated opening values are then committed by a perfectly binding (and extractable) commitment. A much more efficient construction [FLM11] has been recently proposed, that is in constant size. The Abdalla *et al.* construction [ABB+13] improves on the Canetti and Fischlin's one, but remains a linear $\mathtt{E}^2$-commitment.

SPHF-Friendly Commitments. As SPHF is defined for a language $\mathcal{L} \subseteq X$. The first property of an SPHF is that, for a word C in $\mathcal{L}$, the hash value can be computed using either a *secret* hashing key hk or a *public* projected key hp together a witness w to the fact that C is indeed in $\mathcal{L}$. However, for a word C not in $\mathcal{L}$, the hash value computed with hk is perfectly random, even knowing hp. The latter property is the so-called *smoothness* property.

A classical language is the language of the valid commitments of a value x. As remarked above, in order to be compatible with an SPHF, and namely to allow the smoothness property, such an E^2-commitment must be robust, as defined in [ABB$^+$13]: while the simulator is able to generate equivocable commitments, that it will be able to later open in any way, the adversary should not be able to generate commitments that are not perfectly binding. Indeed, when one knows that an adversarially-generated commitment C is perfectly binding, and extracts to x' or $\perp$, then one knows that C does not belong to the language $\mathcal{L}$ of the valid commitments of x, and then one can apply the smoothness property. Therefore, this is enough to reveal random keys (for PAKE) to the adversary, for the commitments it generated in a wrong way, since the hash values look random, as produced by the ideal functionality. However, it is not possible to explain where the random key produced by the ideal functionality comes from, even if it is indistinguishable from the random hash value. Hence, in case of corruption, hk cannot be revealed, and thus has to be securely erased.

As a consequence, to achieve security without requiring reliable erasures, one should be able to explain any random value as the hash value of a word C not in the language. For that, since hp is already published, one has to find an hk that is compatible with both hp and the hash value.

1.4 Our Contributions

The main contributions of this paper are the following. First, in Section 3, we introduce our new notion of explainable projective hash functions (EPHFs), which gives us the capability to generate hash keys that are compatible with the view of the adversary in case of corruption. In addition to defining it, we also propose a generic way of building these primitives.

Second, we propose new constructions of semi-adaptive OT and PAKE schemes without erasures in Section 4. Our new protocols are very similar to the adaptively UC-secure constructions in [ABB$^+$13], except that the underlying commitment schemes are adaptively *without erasures* and the corresponding SPHF is *explainable*.

Third, we propose in Section 5 a new SPHF-friendly commitment, that is significantly more efficient than the one by Abdalla *et al.* [ABB$^+$13]. This immediately leads to the most efficient one-round PAKE and OT protocols, secure in the UC framework, with erasures, but under the plain DDH assumption (and thus without any pairings, contrarily to [ABB$^+$13]).

Finally, in Section 6, we show how to efficiently enhance these protocols with *non-committing encryption* (NCE) in order to achieve adaptive security. In particular, we propose several adaptive versions of our semi-adaptive OT and PAKE protocols, yielding different trade-offs in terms of communication complexity and number of rounds. In each case, at least one of our new protocols outperforms existing ones. Due to space restrictions, complete proofs and some details were postponed to the appendix.

1.5 Related Work

Hash proof systems were introduced by Cramer and Shoup [CS98, CS02] as a means to design chosen-ciphertext-secure public-key encryption schemes. Variants of smooth projective hash functions (SPHFs) have thereafter been proposed in [GL03, KV11, BBC$^+$13b] to build more efficient password-authenticated key-exchange (PAKE) protocols both in the standard model and in the UC framework (please refer to [BBC$^+$13b, BBC$^+$13a] for a more precise characterization of these variants). More recently, Abdalla *et al.* [ABB$^+$13] further improved these results by providing new constructions of SPHFs and commitment schemes and used them to build quite efficient PAKE and

OT protocols with adaptive security in the UC framework, but under the assumption of *reliable erasures*. Removing the need for reliable erasures is the main goal of this work.

Our notion of *explainable projective hash functions* (EPHFs) is close to the notion of *dual projective hashing* (DPH) [Wee12]: the latter also provides a trapdoor that allows us to compute the hashing key that leads to any hash value for a word outside the language. But EPHFs can be seen as the *dual* of DPH:

- In DPH, the word outside the language is honestly generated, but not hp;
- In EPHFs, hp is honestly generated, but not the word outside the language.

In both cases, a trapdoor allows the recovery of an hk that is compatible with hp and H, for any H.

Password-Authenticated Key Exchange (PAKE) protocols were first proposed by Bellovin and Merritt [BM92] as key exchange protocols where the authentication is done using a simple password, subject to exhaustive search. Since then, several PAKE protocols have been proposed in the random-oracle model (e.g., [BPR00, BMP00, AP05]), in the standard model (e.g, [KOY01, GL03, KV11, BBC+13b]), and in the plain model (e.g., [GL01, GJO10]). Among those not relying on ideal models, the most efficient constructions are those based on the Gennaro-Lindell (GL) framework [GL03], which itself is a generalization of the PAKE construction by Katz, Ostrovsky, and Yung (KOY) [KOY01] based on the Cramer-Shoup encryption scheme [CS98].

The first ideal functionality for PAKE was proposed by Canetti *et al.* [CHK+05], together with an efficient protocol based on the GL/KOY methodology [GL03]. Their construction, however, was not known to be secure against adaptive adversaries. Besides the generic but inefficient construction of Barak *et al.* [BCL+05], the first reasonably practical adaptively secure PAKE scheme was proposed by Abdalla *et al.* [ACP09]. As the Canetti *et al.* [CHK+05] PAKE scheme, it also followed the GL/KOY methodology, but used a different commitment scheme, the one by Canetti and Fischlin in [CF01]. Their scheme has been recently improved by Abdalla *et al.* [ABB+13], using a more efficient commitment scheme and an appropriate SPHF. In particular, the latter protocol is one-round, which means that the two players can independently send their flows.

Oblivious Transfer (OT) was introduced in 1981 by Rabin [Rab81] as a way to allow a receiver to get exactly one out of k messages sent by another party, the sender. In these schemes, the receiver should be oblivious to the other values, and the sender should be oblivious to which value was received. Several concrete constructions have been proposed in the UC framework [NP01, CLOS02], some of which being quite efficient [HK07, PVW08, CKWZ13, ABB+13]. Among those that are secure against adaptive corruptions in the global single CRS model, the scheme by Abdalla *et al.* in [ABB+13] seems to be the most efficient. However, it requires *reliable erasures*.

2 Definitions

2.1 Notations

As usual, all the players and the algorithms will be possibly probabilistic and stateful. Namely, adversaries can keep a state st during the different phases, and we denote $\stackrel{\$}{\leftarrow}$ the outcome of a probabilistic algorithm or the sampling from a uniform distribution. For example, $\mathcal{A}(x;r)$ will denote the execution of $\mathcal{A}$ with input x and random tape r. For the sake of clarity, sometimes, the latter random tape will be dropped, with the notation $\mathcal{A}(x)$.

2.2 Smooth Projective Hash Functions

Projective hashing was first introduced by Cramer and Shoup [CS02]. Here we use the formalization of SPHF from [BBC+13b]: Let $\mathcal{X}$ be the domain of the hash functions and let $\mathcal{L}$ be a certain subset of this domain (a language). A key property is that, for a word C in $\mathcal{L}$, the hash value can be computed by using either a *secret* hashing key hk or a *public* projection key hp but with a witness w of the fact that C is indeed in $\mathcal{L}$:

- $\mathsf{HashKG}(\mathcal{L})$ generates a hashing key hk for the language $\mathcal{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ derives the projection key hp, possibly depending on the word C;
- $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ outputs the hash value from the hashing key, for any word $C \in \mathcal{X}$;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w)$ outputs the hash value from the projection key hp, and the witness w, for a word $C \in \mathcal{L}$.

The set of hash values is called the *range* of the SPHF and is denoted Π.

On the one hand, the *correctness* of the SPHF assures that if $C \in \mathcal{L}$ with w a witness of this fact, then $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin \mathcal{L}$, $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ is *statistically* indistinguishable from a random element, even knowing hp.

As in [BBC+13b], we focus on SPHFs for languages of commitments, whose corresponding plaintexts verify some relations, and even more specifically here equal to some value aux. The languages are denoted $\mathcal{L}_{\mathsf{full\text{-}aux}}$, where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$, and crs is the common reference string of the commitment. For some applications, such as PAKE, hk and hp have to be independent of aux, since aux is a secret (the password in case of PAKE). For the sake of simplicity, since we can efficiently achieve it, we restrict HashKG and ProjKG not to use the parameter aux, but just crs (instead of full-aux). But note that this is a stronger restriction than required for our purpose, since one can use aux without leaking any information about it; and some of our applications such as OT do not require aux to be private at all. But, this is not an issue, since none of our SPHFs uses aux.

If HashKG and ProjKG do not depend on C and verify a slightly stronger smoothness property (called adaptive smoothness, which holds even if C is chosen after hp), we say the SPHF is a KV-SPHF. Otherwise, it is said to be a GL-SPHF. See [BBC+13b] for details on GL-SPHF and KV-SPHF and language definitions.

2.3 SPHF-Friendly Commitment Schemes

In this section, we briefly sketch the definition of SPHF-friendly commitment schemes we will use in this paper (more details are given in Appendix A.3). This is a slightly stronger variant of the one in [ABB+13], since it requires an additional polynomial-time algorithm C.IsBinding. But the construction in [ABB+13] still satisfies it. This is a commitment scheme that is both equivocable and extractable. It is defined by the following algorithms: $\mathsf{C.Setup}(1^{\mathfrak{K}})$ generates the global parameters, passed through the global CRS crs to all other algorithms, while $\mathsf{C.SetupT}(1^{\mathfrak{K}})$ is an alternative that additionally outputs a trapdoor τ; $\mathsf{C.Com}^\ell(\boldsymbol{M})$ outputs a pair (C, δ), where C is the commitment of the message $\boldsymbol{M}$ for the label ℓ, and δ is the corresponding opening data, used by $\mathsf{C.Ver}^\ell(C, \boldsymbol{M}, \delta)$ to check the correct opening for C, $\boldsymbol{M}$ and ℓ. It always outputs 0 (false) on $\boldsymbol{M} = \perp$. The trapdoor τ can be used by $\mathsf{C.Sim}^\ell(\tau)$ to output a pair (C, eqk), where C is a commitment and eqk an equivocation key that is later used by $\mathsf{C.Open}^\ell(\mathsf{eqk}, C, \boldsymbol{M})$ to open C on any message $\boldsymbol{M}$ with an appropriate opening data δ. The trapdoor τ can also be used by $\mathsf{C.Ext}^\ell(\tau, C)$ to output the committed message $\boldsymbol{M}$ in C, or $\perp$ if the commitment is invalid. Eventually, the trapdoor τ also allows $\mathsf{C.IsBinding}^\ell(\tau, C, \boldsymbol{M})$ to check whether the commitment C is binding to the message $\boldsymbol{M}$ or not: if there exists $\boldsymbol{M}' \neq \boldsymbol{M}$ and δ', such that $\mathsf{C.Ver}^\ell(C, \boldsymbol{M}', \delta') = 1$, then it outputs 0.

All these algorithms should satisfy some correctness properties: all honestly generated commitments open and verify correctly, can be extracted and are binding to the committed value, while the simulated commitments can be opened on any message.

Then, some security guarantees should be satisfied as well, when one denotes the generation of fake commitments $(C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.SCom}^\ell(\tau, \boldsymbol{M})$, computed as $(C, \mathsf{eqk}) \stackrel{\$}{\leftarrow} \mathsf{C.Sim}^\ell(\tau)$ and then $\delta \leftarrow \mathsf{C.Open}^\ell(\mathsf{eqk}, C, \boldsymbol{M})$:

- *Setup Indistinguishability*: one cannot distinguish the CRS generated by C.Setup from the one generated by C.SetupT;
- *Strong Simulation Indistinguishability*: one cannot distinguish a real commitment (which is generated by C.Com) from a fake commitment (generated by C.SCom), even with oracle access to

the extraction oracle ($\mathsf{C.Ext}$), the binding test oracle ($\mathsf{C.IsBinding}$), and to fake commitments (using $\mathsf{C.SCom}$);
- *Robustness*: one cannot produce a commitment and a label that extracts to $\boldsymbol{M}$ (possibly $\boldsymbol{M} = \perp$) such that $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M}) = 0$, even with oracle access to the extraction oracle ($\mathsf{C.Ext}$), the binding test oracle ($\mathsf{C.IsBinding}$), and to fake commitments (using $\mathsf{C.SCom}$).

Note that, for excluding trivial attacks, on fake commitments, the extraction oracle outputs the $\mathsf{C.SCom}$-input message and the binding test oracle accepts for the $\mathsf{C.SCom}$-input message too. Finally, an SPHF-friendly commmitment scheme has to admit an SPHF for the following language: $\mathcal{L}_{\mathsf{full\text{-}aux}} = \{(\ell, C) \mid \exists \delta,\ \mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta) = 1\}$, where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$ and $\boldsymbol{M} = \mathsf{aux}$.

Basically, compared to the original definition in [ABB+13], the main difference is that it is possible to check in polynomial time (using $\mathsf{C.IsBinding}$) whether a commitment is perfectly binding or not, i.e., does not belong to any $\mathcal{L}_{(\mathsf{crs}, \boldsymbol{M}')}$ for $\boldsymbol{M}' \neq \boldsymbol{M}$, where $\boldsymbol{M}$ is the value extracted from the commitment via $\mathsf{C.Ext}$. In addition, in the games for the strong simulation indistinguishability and the robustness, the adversary has access to this oracle $\mathsf{C.IsBinding}$.

Finally, for our PAKE protocols, as in [ABB+13], we need another property called strong pseudo-randomness. This property is a strong version of the pseudo-randomness property. However, while the latter is automatically verified by any SPHF-friendly commitment scheme, the former may not, because of an additional information provided to the adversary. But, it is verified by the SPHF-friendly commitment scheme in [ABB+13] and by our new commitment scheme introduced in Section 5, which is the most efficient known so far, based on the plain DDH.

2.4 SPHF-Friendly Commitment Schemes without Erasures

We will say that an SPHF-friendly commitment scheme is *without erasures* if this is an SPHF-friendly commitment scheme where δ (and thus the witness) just consists of the random coins used by the algorithm $\mathsf{C.Com}$. Then, an SPHF-friendly commitment scheme without erasures yields directly a commitment scheme that achieves UC-security without erasures.

We remark that slight variants of the constructions in [ACP09, ABB+13] are actually *without erasures*, as long as it is possible to sample obliviously an element from a cyclic group. To make these schemes without erasures, it is indeed sufficient to change the commitment algorithm $\mathsf{C.Com}$ to generate random ciphertexts (with elements obliviously sampled from the corresponding cyclic groups) instead of ciphertexts of 0, for the unused ciphertexts (i.e., the ciphertexts $b_{i,\overline{M}_i}$, for [ABB+13], using the notations in that paper). This does not change anything else, since these ciphertexts are not used in the verification algorithm $\mathsf{C.Ver}$.

In the sequel, all SPHF-friendly commitment schemes are assumed to be *without erasures*. Variants of [ACP09, ABB+13] are possible instantiations, but also our quite efficient constructions presented in Section 5 and Appendix C.

3 Explainable Projective Hashing

3.1 Definition

Let us first suppose there exists an algorithm Setup which takes as input the security parameter $\mathfrak{K}$ and outputs a CRS crs together with a trapdoor τ. In our case Setup will be $\mathsf{C.SetupT}$, and the trapdoor τ will be the commitment trapdoor, which may need to be slightly modified, as we will see in our constructions. This modification generally roughly consists in adding the discrete logarithms of all used elements and is possible with most concrete commitment schemes.

An *explainable projective hashing* (EPH) is a projective hashing with the following additional property: it is possible to generate a random-looking projection key hp, and then receive some hash value H, some value aux and some word $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$, and eventually generate a valid hashing key hk which corresponds to hp and H, as long as we know τ. In other words, it is possible to generate hp

and then "explain" any hash H for a word outside the language $\mathcal{L}_{\mathsf{full\text{-}aux}}$, by giving the appropriate hk.

Whereas DPH [Wee12] implies a weak version of smoothness, our notion of EPH implies the usual notion of smoothness, and is thus stronger than SPHF. Then, an EPHF can be either a KV-EPHF or a GL-EPHF, depending on whether the word C is known when hp is generated.

An Explainable Projective Hash Function (EPHF) is defined by the following algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the global CRS crs or $\mathsf{full\text{-}aux}$ to all the other algorithms, plus a trapdoor τ;
- HashKG, ProjKG, Hash, and $\mathsf{ProjHash}$ behave as for a classical SPHF;
- $\mathsf{SimKG}(\mathsf{crs}, \tau, C)$ outputs a projection key hp together with an explainability key expk;
- $\mathsf{Explain}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, H, \mathsf{expk})$ outputs an hashing key hk corresponding to hp, $\mathsf{full\text{-}aux}$, C, and H.

It must verify the following properties, for any $(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}})$:

- *Explainability Correctness.* For any aux, any $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$ and any hash value H, if $(\mathsf{hp}, \mathsf{expk}) \stackrel{\$}{\leftarrow} \mathsf{SimKG}(\mathsf{crs}, \tau, C)$ and $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, H, \mathsf{expk})$, then $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}, C)$ and $H = \mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C)$;
- *Indistinguishability.* As for smoothness, we consider two types of indistinguishability:
  - GL-indistinguishability: a GL-EPHF is ε-indistinguishable, if for any aux and any $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$, the two following distributions are ε-close:
$$\{(\mathsf{hk}, \mathsf{hp}) \mid H \stackrel{\$}{\leftarrow} \Pi; (\mathsf{hp}, \mathsf{expk}) \stackrel{\$}{\leftarrow} \mathsf{SimKG}(\mathsf{crs}, \tau, C); \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, H, \mathsf{expk})\}$$
$$\{(\mathsf{hk}, \mathsf{hp}) \mid \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}, C)\}.$$
  - KV-indistinguishability: a KV-EPHF is ε-indistinguishable, if for any aux and any function f from the set of projection keys to $\mathcal{X} \setminus \mathcal{L}_{\mathsf{full\text{-}aux}}$, the two following distributions are ε-close:
$$\{(\mathsf{hk}, \mathsf{hp}) \mid H \stackrel{\$}{\leftarrow} \Pi; (\mathsf{hp}, \mathsf{expk}) \stackrel{\$}{\leftarrow} \mathsf{SimKG}(\mathsf{crs}, \tau, \perp); \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, \mathsf{full\text{-}aux}, f(\mathsf{hp}), H, \mathsf{expk})\}$$
$$\{(\mathsf{hk}, \mathsf{hp}) \mid \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}, \perp)\}.$$

Explainable Projective Hashing with Hint. An EPHF with hint is similar to an EPHF except the algorithm $\mathsf{Explain}$ takes an additional argument hint, an hint that verifies some implicit relation with H. For example, hint can be the discrete logarithm of H. The indistinguishability property is similar except it only has to hold when hint is a valid hint for H.

In the following, we will provide a generic construction of EPHF from any SPHF, and in Appendix E, we propose a more involved construction of EPHF *with hint*. This weaker version is enough for OT protocols.

3.2 Generic Construction

Generic Construction of GL-EPHF. Let us consider a GL-SPHF for which:

1. for any hashing key hk and associated projection key hp, it is possible to draw random hk' corresponding to hp, such that the hash value of a word $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$ under hk' is uniform. More precisely, we suppose there exists a randomized algorithm $\mathsf{InvProjKG}$, which takes as input τ, a hashing key hk, crs, and possibly a word $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$, and outputs a random hashing key hk', verifying $\mathsf{ProjKG}(\mathsf{hk}', \mathsf{crs}, C) = \mathsf{hp}$. For any $(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}})$, for any aux, for any $C \notin \mathcal{L}_{\mathsf{full\text{-}aux}}$, with overwhelming probability over $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$, the two following distributions are supposed to be identical (or ε-close, with ε negligeable in $\mathfrak{K}$):
$$\{H \mid \mathsf{hk}' \stackrel{\$}{\leftarrow} \mathsf{InvProjKG}(\tau, \mathsf{hk}, \mathsf{crs}, C); H \leftarrow \mathsf{Hash}(\mathsf{hk}', \mathsf{full\text{-}aux}, C)\} \qquad \{H \mid H \stackrel{\$}{\leftarrow} \Pi\}.$$
   This property can be seen as a strong version of smoothness.

2. there exists a parameter ν polynomial in $\log \mathfrak{K}$ and a randomness extractor $\mathsf{Extract}$ with range $\{0,1\}^\nu$, such that the two following distributions are ε-close (with ε negligeable in $\mathfrak{K}$):

$$\{\mathsf{Extract}(H) \mid H \stackrel{\$}{\leftarrow} \Pi\} \qquad\qquad \{H \mid H \stackrel{\$}{\leftarrow} \{0,1\}^\nu\}.$$

Details on the randomness extractor can be found in Appendix A.2. But either a deterministic extractor exists for Π, which is possible for many cyclic groups [CFPZ09], or one uses a probabilistic extractor with an independent random string in the CRS.

Then, if the hash values H computed by Hash or $\mathsf{ProjHash}$ are replaced by $\mathsf{Extract}(H)$, the resulting SPHF is a $\mathsf{GL\text{-}EPHF}$. Indeed, if $\mathsf{SimKG}(\mathsf{crs}, \tau, C)$ just generates $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$ and $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}, C)$, and outputs hp and $\mathsf{expk} = (\tau, \mathsf{hk})$. Then, $\mathsf{Explain}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, H, \mathsf{expk})$ just runs $\mathsf{hk}' \stackrel{\$}{\leftarrow} \mathsf{InvProjKG}(\tau, \mathsf{hk}, \mathsf{crs}, C)$ many times until it finds hk' such that $\mathsf{Hash}(\mathsf{hk}', \mathsf{full\text{-}aux}, C) = H$. Thanks to the above properties, it should find a valid hk' after about 2^ν runs. Since ν is polynomial in $\log \mathfrak{K}$, the resulting algorithm $\mathsf{Explain}$ is polynomial in $\mathfrak{K}$.

Actually, ν will determine the tightness of the proof. In all comparisons in this article, we will use $\nu = 1$, which hinders performances of our scheme; but our schemes are still very efficient. In practice, to gain constant factors, it would be advisable to use a greater ν, and thus larger blocks. Finally, the range of the EPHF can be easily extended just by using multiple copies of the EPHF: for a range of ν', hk becomes a tuple of $\lceil \nu'/\nu \rceil$ original hashing keys, the same for hp and H.

Application to SPHFs Built Using the Generic Framework of [BBC+13b]. Although the first property may seem really restrictive, most (if not all) current SPHFs verify it if τ is chosen correctly. In particular, SPHFs built using the generic framework of [BBC+13b] verify it, basically as long as τ contains the discrete logarithms of all elements.

Generic Construction for KV-EPHF. In the previous generic construction, we get a $\mathsf{KV\text{-}EPHF}$, if the security property related to $\mathsf{InvProjKG}$ holds even if C can depend on hp. More precisely, we want the following property: For any $(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}})$, for any aux, for any function f from the set of projection keys to $\mathcal{X} \setminus \mathcal{L}_{\mathsf{full\text{-}aux}}$, with overwhelming probability over $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$, with $\mathsf{hp} \leftarrow \mathsf{ProjHash}(\mathsf{hk}, \mathsf{crs}, \perp)$, the two following distributions are supposed to be identical (or ε-close, with ε negligeable in $\mathfrak{K}$):

$$\{H \mid \mathsf{hk}' \stackrel{\$}{\leftarrow} \mathsf{InvProjKG}(\tau, \mathsf{hk}, \mathsf{crs}, \perp); H \leftarrow \mathsf{Hash}(\mathsf{hk}', \mathsf{full\text{-}aux}, f(\mathsf{hp}))\} \qquad \{H \mid H \stackrel{\$}{\leftarrow} \Pi\}.$$

4 Semi-Adaptive OT and PAKE without Erasures

In this section, we propose two new OT and PAKE protocols that are UC-secure against semi-adaptive adversaries, but without requiring reliable erasures. The security proofs can be found in Appendix B. Actually, these protocols are very similar to the UC-secure constructions in [ABB+13], except that the SPHF-friendly commitment scheme has to be *without erasures* and the SPHF has to be *explainable*. However, the proof is more complicated.

4.1 Semi-Adaptivity

The semi-adaptive setting has been introduced in [GWZ09], for two-party protocols when channels are authenticated: the adversary is not allowed to corrupt any player if the two players were honest at the beginning of the protocol. When channels are not authenticated, as for PAKE, we restrict the adversary not to corrupt a player P_i if an honest flow has been sent on its behalf, and it has been received by P_j, without being altered. In addition to those restrictions on the adversary, there are also some restrictions on the simulator, which we do not recall here due to lack of space.

4.2 Oblivious Transfer

The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Appendix A.4. It is inspired from [CKWZ13]. In Figure 1, we describe a 2-round 1-out-of-k OT for ν_m-bit messages, that is UC-secure against semi-adaptive adversaries. It can be built from any SPHF-friendly commitment scheme, admitting a $\mathsf{GL\text{-}EPHF}$, with range $\Pi = \{0,1\}^{\nu_m}$, for the language: $\mathcal{L}_{\mathsf{full\text{-}aux}} = \{(\ell, C) \mid \exists\delta, \mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta)\} = 1$, where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$ and $\boldsymbol{M} = \mathsf{aux}$.

In case of corruption of the database (sender) after it has sent its flow, since we are in the semi-adaptive setting, the receiver was already corrupted and thus the index s was known to the simulator. The latter can thus generate "explainable" hp_t for all $t \neq s$, so that when the simulator later learns the messages m_t, it can explain hp_t with appropriate hk_t. Erasures are no longer required, contrarily to [ABB+13].

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{C.Setup}(1^{\mathfrak{K}})$.
Index query on s:
1. P_j computes $(C, \delta) \xleftarrow{\$} \mathsf{C.Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
2. P_j sends C to P_i

Database input $(m_1, \ldots, m_k)$:
1. P_i computes $\mathsf{hk}_t \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs})$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, \mathsf{crs}, (\ell, C))$,
   $K_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, (\mathsf{crs}, t), (\ell, C))$, and $M_t \leftarrow K_t \operatorname{xor} m_t$, for $t = 1, \ldots, k$
2. P_i sends $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$

Data recovery:
Upon receiving $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$, P_j computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (\mathsf{crs}, s), (\ell, C), \delta)$ and gets $m_s \leftarrow K_s \operatorname{xor} M_s$.

Fig. 1. UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment for Semi-Adaptive Adversaries

The restriction that Π has to be of the form $\{0,1\}^{\nu_m}$ is implicit in [ABB+13]. Any SPHF can be transformed to an SPHF with range Π of the form $\{0,1\}^{\nu_m}$, using a randomness extractor, as long as the initial range is large enough. However, this is not the case for EPHF, since the extractor may not be reversible. That is why we need to make this assumption on Π explicit.

4.3 Password-Authenticated Key Exchange

The ideal functionality of a Password-Authenticated Key Exchange (PAKE) proposed in [CHK+05] is depicted in Appendix A.4. In Figure 2, we describe a one-round PAKE that is UC-secure against semi-adaptive adversaries. It can be built from any SPHF-friendly commitment scheme, admitting a $\mathsf{KV\text{-}EPHF}$ with strong pseudo-randomness, with range $\Pi = \{0,1\}^{\mathfrak{K}}$.

Again, thanks to the explainability property, it is possible to generate the hashing key that explains the session key provided by the ideal functionality, when the second player gets corrupted: since a first player was already corrupted, the simulator has already extracted the tentative password. In case of good guess by the adversary, the simulator can choose the key, that is thus easy to explain. However, in case of a bad guess by the adversary, the session key is randomly chosen by the functionality. But the simulator knows that the commitment is not in the right language, and so the projection key can be made explainable.

5 New SPHF-Friendly Commitment Scheme

In this section, we present our new efficient SPHF-friendly commitment scheme under the plain DDH. Due to lack of space, we only give an overview of the scheme and a comparison with previous SPHF-friendly commitment schemes. Details are left to Appendix C.

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{C.Setup}(1^{\mathfrak{K}})$. Only protocol execution by P_i is described. The one by P_j is symmetrical.
Protocol execution by P_i with π_i:
1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(L)$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathsf{crs}, \perp)$
2. P_i computes $(C_i, \delta_i) \xleftarrow{\$} \mathsf{C.Com}^{\ell_i}(\pi_i)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i sends hp_i, C_i to P_j

Key computation: Upon receiving hp_j, C_j from P_j
1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathsf{crs}, \pi_i), (\ell_i, C_i), \delta_i)$ and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathsf{crs}, \pi_i), (\ell_j, C_j))$ with $\ell_j = (\mathsf{sid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{SK}_i \leftarrow H'_i \operatorname{xor} H_j$.

Fig. 2. One-Round UC-Secure PAKE from an SPHF-Friendly Commitment for Semi-Adaptive Adversaries

5.1 Scheme

Basic Idea of Previous Schemes. The basic idea of our scheme is similar to the one in [CF01, CLOS02, ACP09, ABB+13]: to commit to some bit b, a user essentially generates some element P and two words C_0 and C_1 such that $C_b \in L_{P,b}$, where $L_{P,0}$ and $L_{P,1}$ are two languages[1]. To open the commitment, the user just gives the random coins used to generate C_b, which proves that $C_b \in L_{P,b}$.

The two words also have to be related, in such a way that an adversary cannot generate P and two words C_0 and C_1 such that $C_0 \in L_{P,0}$ and $C_1 \in L_{P,1}$. However, when in possession of a given trapdoor, we can compute such words, which enables us to generate simulated commitments which can later be opened to the bit of our choice. This property is crucial for robustness, since it ensures that a commitment produced by an adversary is necessarily perfectly binding.

In addition, using another trapdoor, it is possible to check whether $C_b \in L_{P,b}$ or not (without knowing the random coins used to generate C_b. This makes the commitment extractable.

For all the previous constructions, to ensure these properties, P was an equivocable commitment of the bit b to be committed, such as the Pedersen commitment [Ped91] in [ACP09] or the Haralambiev commitment [Har11] in [ABB+13], and $L_{P,0}$ and $L_{P,1}$ were the languages of ciphertexts (for an IND-CCA encryption scheme such as Cramer-Shoup [CS98]) of a valid opening of P for 0 and 1 respectively. The binding property of the commitment P was used to prove an adversary could not generate P together with two words $C_0 \in L_{P,0}$ and $C_1 \in L_{P,1}$.

Unfortunately, the most efficient instantiation to date of this idea, namely the commitment of Abdalla *et al.* [ABB+13], requires an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, due to the use of the Haralambiev commitment, and $8m$ elements in $\mathbb{G}_1$ (for the two Cramer-Shoup ciphertexts) and 1 element in $\mathbb{G}_2$ (for the Haralambiev commitment), for each bit.

Our New Scheme. Here, we improve on this construction in the following way: C_0 and C_1 are now similar to Cramer-Shoup ciphertexts but without the part depending on the plaintext. To ensure that no adversary can generate two words $C_0 \in L_{P,0}$ and $C_1 \in L_{P,1}$, we just ensure that the product of the first elements (denoted $u_{i,0}$ and $u_{i,1}$ for the i-th bit) of C_0 and C_1 be some fixed element T. An additional "randomization" using some elements denoted e_{i,M_i} is necessary to prevent the user from distinguishing simulated commitments from normal ones. The last parts of C_0 and C_1 are adapted consequently.

But even with this randomization, since we do not need the part of the Cramer-Shoup ciphertext with the plaintext element nor a Pedersen-like commitment, our scheme is much more efficient, as shown in Section 5.2.

More precisely, $\mathsf{C.SetupT}(1^{\mathfrak{K}})$ generates a cyclic group $\mathbb{G}$ of order p, three generators g, h, $\hat{h}$, a tuple $(\alpha, \beta, \gamma, \alpha', \beta', \gamma') \leftarrow \mathbb{Z}_p^6$, and H is a random collision-resistant hash function from some family $\mathcal{H}$. It then computes the tuple $(c = g^{\alpha}\hat{h}^{\gamma}, d = g^{\beta}h^{\gamma}, c' = g^{\alpha'}\hat{h}^{\gamma'}, d' = g^{\beta'}h^{\gamma'})$. It also generates a random scalar $t \xleftarrow{\$} \mathbb{Z}_p$ and sets $T = g^t$. The CRS crs is set as $(g, h, \hat{h}, H, c, d, c', d', T)$ and the trapdoor τ is the decryption key $(\alpha, \alpha', \beta, \beta', \gamma, \gamma')$ (a.k.a., extraction trapdoor) together with t (a.k.a., equivocation trapdoor).

[1] These languages have nothing to do with the languages of SPHF, that is why the notation is different

Table 1. Comparison with existing non-interactive UC-secure commitments with a single global CRS (m = bit-length of the committed value, $\mathfrak{K}$ = security parameter)

	SPHF Friendly	W/o Erasure	C size	δ size	hp size KV / GL	Assumption
[CF01]	no	yes	$9m \times \mathbb{G}$	$2m \times \mathbb{Z}_p$	–	Plain DDH
[ACP09]	yes	yes	$(m + 16m\mathfrak{K}) \times \mathbb{G}$	$2m\mathfrak{K} \times \mathbb{Z}_p$	– / $(3m + 2) \times \mathbb{G} + (\mathbb{Z}_p)^{\mathrm{a}}$	Plain DDH
[FLM11], 1	no	no	$5 \times \mathbb{G}$	$16 \times \mathbb{G}$	–	DLin
[FLM11], 2	no	no	$37 \times \mathbb{G}$	$3 \times \mathbb{G}$	–	DLin
[ABB+13]	yes	yes	$8m \times \mathbb{G}_1 + m \times \mathbb{G}_2$	$m \times \mathbb{Z}_p$	$2m \times \mathbb{G}_1$ / $\mathbb{G}_1 + (\mathbb{Z}_p)^{\mathrm{a}}$	SXDH
this paper	yes	yes	$7m \times \mathbb{G}$	$2m \times \mathbb{Z}_p$	$4m \times \mathbb{G}$ / $2 \times \mathbb{G} + (\mathbb{Z}_p)^{\mathrm{a}}$	Plain DDH

[a] this $\mathbb{Z}_p$ element may only be $\mathfrak{K}$-bit long and is useless when $m = 1$.

Then, the commitment of a message $\boldsymbol{M} = (M_i)_i \in \{0,1\}^m$ under a label ℓ, works as follows, where $\overline{M}_i$ denotes $1 - M_i$: For $i = 1, \dots, m$, it chooses two random scalars $r_{i,M_i}, s_{i,M_i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sets:

$$\begin{array}{llll} e_{i,M_i} = g^{r_{i,M_i}} & u_{i,M_i} = g^{s_{i,M_i}} & v_{i,M_i} = \hat{h}^{r_{i,M_i}} h^{s_{i,M_i}} & w_{i,M_i} = (c^{r_{i,M_i}} \cdot d^{s_{i,M_i}}) \cdot (c'^{r_i} d'^{s_{i,M_i}})^{\xi} \\ e_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{G} & u_{i,\overline{M}_i} = T/u_{i,M_i} & v_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{G} & w_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \end{array}$$

with $\xi = H(\ell, (e_{i,b}, u_{i,b}, v_{i,b})_{i,b})$. The commitment is $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b} \in \mathbb{G}^{8m}$, while the opening information is the $2m$-tuple $\delta = (r_{i,M_i}, s_{i,M_i})_i \in \mathbb{Z}_p^{2m}$.

For each i, $(e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})$ corresponds to the word C_b. The language L_b is just the set of such tuples as generated for $b = M_i$ in the commitment procedure, described above. The binding property comes from the fact that $u_{i,0} \cdot u_{i,1}$ has to be equal to T. By knowing t, the discrete logarithm of T in base g, it is therefore easy to generate an equivocable commitment.

It remains to show how to extract a commitment. For that, we can roughly show that with high probability, $(e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})$ is generated as in the commitment procedure, if and only if:

$$w_{i,b} = e_{i,b}^{\alpha+\xi\alpha'} \cdot u_{i,b}^{\beta+\xi\beta'} \cdot v_{i,b}^{\gamma_b+\xi\gamma'_b}.$$

This check is similar to the one used to check the validity of Cramer-Shoup ciphertexts.

For the reader acquainted with 2-universal hash proof systems [CS02], another way to look at this test (and at our commitment scheme in general) is the following: w_{i,M_i} is the hash value of the tuple $(g, e_{i,M_i}, u_{i,M_i}, v_{i,M_i})$ under a 2-universal SPHF with hashing key $(\alpha, \beta, \gamma, \alpha', \beta', \gamma')$ and projection key (c, d, c', d'). This hash value enables us to "prove" that $v_{i,b} = \hat{h}^{\log_g e_{i,b}} h^{\log_g u_{i,b}}$. To construct a KV-SPHF and a GL-SPHF for this commitment, we can use the generic framework in [BBC+13b]. Details can be found in C.2.

5.2 Complexity and Comparison

In our new scheme, we remark that $u_{i,1}$ can be computed from $u_{i,0}$ as $u_{i,1} = T/u_{i,0}$. So, in the sequel, we suppose that $u_{i,1}$ is not a part of the commitment, when we analyze our commitment complexity. However, for the sake of simplicity, we keep $u_{i,1}$ in the commitments in our proofs.

Table 1 compares our new schemes with existing non-interactive UC-secure commitments with a single global CRS. Our scheme is the most efficient SPHF-friendly scheme regarding the commitment and decommitment size. In addition, it is secure under plain DDH. Its projection key is slightly longer than the projection key in [ABB+13], and in cases were numerous projection keys are required and if SXDH is deemed acceptable, using the commitment scheme in [ABB+13] may lead to more efficient schemes. Details on the comparison can be found in Appendix C.7.

6 Adaptive OT and PAKE

As explained in [GWZ09], one can transform any semi-adaptive protocols into adaptive ones by sending all the flows through secure channels. Such secure channels can be constructed using non-committing encryption (NCE) [CFGN96,DN00,Bea97a,CDMW09]. However, even the most efficient instantiation of NCE [CDMW09] requires $8\nu_{\mathsf{NCE}}\mathfrak{K}$ group elements to send ν_{NCE} bits securely, with ElGamal encryption scheme as (trapdoor) simulatable encryption scheme. If ν_{NCE} is $\Omega(\mathfrak{K})$, this can be reduced to about $320\nu_{\mathsf{NCE}}$ group elements.

In this section, we propose several adaptive versions of our semi-adaptive OT and PAKE protocols. Some are optimized for the number of rounds, while others are optimized for the communication complexity. In each case, at least one of our new protocols performs better than existing protocols.

6.1 Oblivious Transfer

First Scheme. A first efficient way to construct a bit (i.e., $\nu_m = 1$) 1-out-of-2 OT secure against adaptive adversary consists in applying the generic transformation of Garay *et al.* [GWZ09] to our semi-adaptive OT.

This transformation uses the notion of ℓ-somewhat non-committing encryption scheme. This scheme enables to send securely long messages, but which restricts the non-committing property to the following: it is only possible to produce random coins corresponding to ℓ different messages. Then, to get an adaptive OT from a semi-adaptive OT, it is sufficient to execute the protocol in a 8-somewhat non-committing channel. Indeed, the simulator can send via this channel 8 versions of the transcript of the protocol: depending on which user gets corrupted first and on which were their inputs and outputs. There are two choices of inputs for the sender (the two index queries) and two outputs (the message m_s), hence four choices in total; and there are four choices of inputs for the receiver (the two messages m_0 and m_1). Hence the need for 8 versions.

In [GWZ09], the authors also show how to extend their bit OT based on the DDH version of the static OT of Peikert *et al.* [PVW08] to string OT by repeating the protocol in parallel and adding an equivocable commitment to the index and a zero-knowledge proof to ensure that the sender always uses the same index s. Actually, for both of our instantiations and for the one in [GWZ09], we can do better, just by using the same commitment C to s (in our case) or the same CRS (the one obtained by coin tossing) and the same public key of the dual encryption system (in their case). This enables us to get rid off the additional zero-knowledge proof and can also be applied to the QR instantiation in [GWZ09]. In addition, the commitment C to s (in our case) or the CRS and the public key (in their case) only needs to be sent in the first somewhat non-committing channel.

Finally, if the original semi-adaptive OT is a 1-out-of-k OT (with $k = 2^{\nu_k}$), then we just need to use a 2^{k+1}-somewhat NCE instead of a 8-somewhat NCE encrypt (because there are 2^k possible inputs for the sender, and k possible inputs and 2 possible outputs for the receiver, so $2^k + 2k \leq 2^{k+1}$ possible versions for the transcript).

Second Scheme. Our second scheme can be significantly more efficient than our first one, for several parameter choices. Essentially, it consists in using NCE channels to send km random bits to mask the messages (in case the sender is corrupted first) and $2\nu_k$ random bits to enable the simulator to make the commitment binding to the index s (in case the receiver gets corrupted first). Methods used for this second part are specific to our new SPHF-friendly commitment scheme, but can also be applied to the commitment scheme in [ABB+13].

This protocol can be made a little more efficient by using a greater ν in the EPHFs (see Section 3.2), at the cost of a less tight reduction. Furthermore, if we accept to rely on SXDH, using the commitment in [ABB+13] (whose projection key size is slightly smaller) can further improve the efficiency. However, the bottleneck remains the $km + 2\nu_k$ bits to be sent via NCE. Details can be found in Appendix D.2.

Comparison. In Appendix D.3, we compare our schemes with the DDH-based OT in [GWZ09]. We see that, for every parameters ν_m and k, at least one of our two schemes (if not both) is the most efficient scheme regarding both the number of rounds and the communication complexity.

The exact communication complexity cost depends on the exact instantiation of NCE. But in all cases, at least one of our schemes outperforms existing schemes both in terms of number of bits sent via a NCE channel, and in terms of auxiliary elements (elements which are not directly used by the NCE scheme). In addition, our second scheme always uses the smallest number of auxiliary elements; and it requires $km + 2\nu_k$ bits to be sent via a NCE channel, which is not worse than the $(k+1)m$ bits required by our first scheme, as long as $m \geq 2\nu_k$.

6.2 Password Authenticated Key Exchange

Optimized for Round Complexity. If we apply a slight variant[2] of the transformation of Garay *et al.* to our efficient semi-adaptive PAKE, we get a 3-round PAKE UC-secure against adaptive adversary, without erasures. It requires the use of a NCE channel for approximatively the number of bits sent in the original semi-adaptive PAKE, to mask the complete transcript to ensure the simulator can deal with any corruption. Therefore, this construction is highly inefficient regarding the communication complexity, though maybe not as inefficient as the generic construction of PAKE in [BCL+05].

Optimized for Communication Complexity. We then propose a second construction, much more efficient regarding the communication complexity. This construction is actually generic and can transform any PAKE (for 1-bit passwords) UC-secure against semi-adaptive adversaries into a UC-secure PAKE (for m-bit passwords) UC-secure against adaptive adversaries. The scheme just requires to send $2m + \mathfrak{K}$ bits, via a non-committing encryption scheme, of which $2m$ are used to create m 4-somewhat non-committing encryption schemes used to deal with inputs. The remaining $\mathfrak{K}$ bits are used to mask the final shared key. The scheme is constant-round, if the associated semi-adaptive PAKE is constant-round, and the communication complexity is roughly $4m$ times the one of the semi-adaptive PAKE, plus the cost of the non-committing encryption scheme.

This is much more efficient than the construction of Canetti *et al.* [CDVW12], which requires $2m$ adaptively-secure OT for $\nu_m = \mathfrak{K}$ bits. Indeed, each such OT could be used as a non-committing channel of $\mathfrak{K}$ bits[3], and so their construction requires to send at least $2m\mathfrak{K}$ bits via a non-committing channel.

Acknowledgments

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet

References

ABB+13. M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. SPHF-friendly non-interactive commitments. In *ASIACRYPT 2013, Part I*, *LNCS* 8269, pages 214–234. Springer, December 2013. (Pages 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 16, 18, 25, 30, and 35.)

ACP09. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *CRYPTO 2009*, *LNCS* 5677, pages 671–689. Springer, August 2009. (Pages 4, 6, 10, 11, and 30.)

AP05. M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *CT-RSA 2005*, *LNCS* 3376, pages 191–208. Springer, February 2005. (Page 4.)

BBC+13a. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New smooth projective hash functions and one-round authenticated key exchange. Cryptology ePrint Archive, Report 2013/034, 2013. `http://eprint.iacr.org/2013/034`. (Page 3.)

BBC+13b. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 449–475. Springer, August 2013. (Pages 3, 4, 5, 8, 11, 18, 24, 34, and 35.)

[2] The original transformation implicitly only deals with authenticated channels. But by using one-time signatures or signature schemes in a way similar to the one proposed in [BCL+05], we can make this transformation work with PAKE protocols which use non-authenticated channels.

[3] And actually, as we have seen before, currently, the most efficient OT for $\nu_m = \mathfrak{K}$ bits, even requires $2\mathfrak{K} + 2$ bits sent via a non-committing channel.

BCL+05. B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. In *CRYPTO 2005*, *LNCS* 3621, pages 361–377. Springer, August 2005. (Pages 4, 13, and 34.)

Bea97a. D. Beaver. Commodity-based cryptography (extended abstract). In *29th ACM STOC*, pages 446–455. ACM Press, May 1997. (Page 11.)

Bea97b. D. Beaver. Plug and play encryption. In *CRYPTO'97*, *LNCS* 1294, pages 75–89. Springer, August 1997. (Page 1.)

BM92. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Page 4.)

BMP00. V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *EUROCRYPT 2000*, *LNCS* 1807, pages 156–171. Springer, May 2000. (Page 4.)

BPR00. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, *LNCS* 1807, pages 139–155. Springer, May 2000. (Page 4.)

Can00. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`. (Page 1.)

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Page 1.)

CDMW09. S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT 2009*, *LNCS* 5912, pages 287–302. Springer, December 2009. (Pages 11, 12, 30, and 33.)

CDVW12. R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee. Efficient password authenticated key exchange via oblivious transfer. In *PKC 2012*, *LNCS* 7293, pages 449–466. Springer, May 2012. (Pages 1 and 13.)

CF01. R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO 2001*, *LNCS* 2139, pages 19–40. Springer, August 2001. (Pages 2, 4, 10, 11, and 30.)

CFGN96. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. (Pages 1 and 11.)

CFPZ09. C. Chevalier, P.-A. Fouque, D. Pointcheval, and S. Zimmer. Optimal randomness extraction from a Diffie-Hellman element. In *EUROCRYPT 2009*, *LNCS* 5479, pages 572–589. Springer, April 2009. (Pages 8 and 16.)

CHK+05. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In *EUROCRYPT 2005*, *LNCS* 3494, pages 404–421. Springer, May 2005. (Pages 2, 4, 9, and 18.)

CKWZ13. S. G. Choi, J. Katz, H. Wee, and H.-S. Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In *PKC 2013*, *LNCS* 7778, pages 73–88. Springer, February / March 2013. (Pages 2, 4, 9, and 18.)

CLOS02. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. (Pages 4 and 10.)

CS98. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*, *LNCS* 1462, pages 13–25. Springer, August 1998. (Pages 3, 4, and 10.)

CS02. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, April / May 2002. (Pages 2, 3, 4, 11, and 18.)

DN00. I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO 2000*, *LNCS* 1880, pages 432–450. Springer, August 2000. (Pages 1 and 11.)

FLM11. M. Fischlin, B. Libert, and M. Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In *ASIACRYPT 2011*, *LNCS* 7073, pages 468–485. Springer, December 2011. (Pages 2 and 11.)

GJO10. V. Goyal, A. Jain, and R. Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In *CRYPTO 2010*, *LNCS* 6223, pages 277–294. Springer, August 2010. (Page 4.)

GL01. O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In *CRYPTO 2001*, *LNCS* 2139, pages 408–432. Springer, August 2001. `http://eprint.iacr.org/2000/057`. (Page 4.)

GL03. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, *LNCS* 2656, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. (Pages 2, 3, and 4.)

GMR88. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. (Page 16.)

GWZ09. J. A. Garay, D. Wichs, and H.-S. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO 2009*, *LNCS* 5677, pages 505–523. Springer, August 2009. (Pages 1, 2, 8, 11, 12, 20, 33, and 34.)

Har11. K. Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications*. PhD thesis, New York University, 2011. (Page 10.)

HILL99. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. (Page 16.)

HK07. O. Horvitz and J. Katz. Universally-composable two-party computation in two rounds. In *CRYPTO 2007*, *LNCS* 4622, pages 111–129. Springer, August 2007. (Page 4.)

KOY01. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001*, *LNCS* 2045, pages 475–494. Springer, May 2001. (Page 4.)

KV11. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, March 2011. (Pages 3 and 4.)

MT09. R. Montenegro and P. Tetali. How long does it take to catch a wild kangaroo? In *41st ACM STOC*, pages 553–560. ACM Press, May / June 2009. (Page 36.)

NP01. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Page 4.)

Ped91. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, *LNCS* 576, pages 129–140. Springer, August 1991. (Page 10.)

PVW08. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, *LNCS* 5157, pages 554–571. Springer, August 2008. (Pages 4, 12, and 33.)

Rab81. M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981. (Page 4.)

Wee12. H. Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In *EUROCRYPT 2012*, *LNCS* 7237, pages 246–262. Springer, April 2012. (Pages 4 and 7.)

A Notations

We first recall the classical definitions on distances of distribution, and the notions of success and advantage. We then review the basic cryptographic tools we use along this paper, with the corresponding security notions.

A.1 Distances, Advantage and Success

Statistical Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is also the statistical distance between X_0 and X_1:

$$\mathtt{Dist}(\mathcal{D}_0, \mathcal{D}_1) = \mathtt{Dist}(X_0, X_1) = \sum_{x \in \mathcal{S}} \left| \Pr\left[\, X_0 = x \,\right] - \Pr\left[\, X_1 = x \,\right] \right|.$$

If the statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is less than or equal to ε, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_0$ and $\mathcal{D}_1$ are 0-close, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are perfectly indistinguishable.

Success/Advantage. When one considers an experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A}}(\mathfrak{K})$ in which adversary $\mathcal{A}$ plays a security game $\mathtt{SEC}$, we denote $\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\, \mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A}}(\mathfrak{K}) = 1 \,\right]$ the success probability of this adversary. We additionally denote $\mathsf{Succ}^{\mathsf{sec}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal success any adversary running within time t can get.

When one considers a pair of experiments $\mathsf{Exp}^{\mathsf{sec}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$, for $b = 0, 1$, in which adversary $\mathcal{A}$ plays a security game $\mathtt{SEC}$, we denote $\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\, \mathsf{Exp}^{\mathsf{sec}-0}_{\mathcal{A}}(\mathfrak{K}) = 1 \,\right] - \Pr\left[\, \mathsf{Exp}^{\mathsf{sec}-1}_{\mathcal{A}}(\mathfrak{K}) = 1 \,\right]$ the advantage of this adversary. We additionally denote $\mathsf{Adv}^{\mathsf{sec}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal advantage any adversary running within time t can get.

Computational Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The computational distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is the best advantage an adversary can get in distinguishing X_0 from X_1: $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\, \mathcal{A}(X_0) = 1 \,\right] - \Pr\left[\, \mathcal{A}(X_1) = 1 \,\right]$, and thus $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}, \mathfrak{K})\}$. When the advantage $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(t) \leq \varepsilon$, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

We can note that for two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ that are ε-close, for any t and ε, $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

A.2 Formal Definitions of the Basic Primitives

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions H_k from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^{\mathfrak{K}}$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $H \stackrel{\$}{\leftarrow} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\, H \stackrel{\$}{\leftarrow} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(H) : H(m_0) = H(m_1) \,\right].$$

It is well-known that under the discrete logarithm problem, and thus under the DDH assumption, such collision-resistant hash functions can be built.

Randomness Extractor. A randomness extractor allows to extract uniform bit-strings from high-entropy bit-string sources. The most famous method is provided by the Leftover Hash Lemma [HILL99], which requires the use of universal hash function families. From an additional independent random source to select the hash function, one can extract a bit-string that is almost uniform.

More precisely, by randomly selecting a random function h, from a universal hash function family, in the CRS, from a random variable X with min-entropy m, one can extract k-bit strings that are 2^e-close to uniform by computing $h(X)$,if $k \leq m - 2e + 2$.

In the particular case of cyclic groups, in well-chosen finite fields or elliptic curves, some efficient deterministic extractors, such as the truncation, can be used [CFPZ09].

Signature Schemes and One-Time Signature Schemes. A signature scheme is defined by three algorithms:

– $\mathsf{Sig.KG}(1^{\mathfrak{K}})$ generates a verification key vk together with a signing key sk;
– $\mathsf{Sig.Sign}(\mathsf{sk}, M)$ generates a signatures σ of M;
– $\mathsf{Sig.Verify}(\mathsf{vk}, \sigma, M)$ returns 1 if σ is a valid signature of M; and 0 otherwise.

The basic security notion for signatures is existential unforgeability under chosen-message attacks (defined in [GMR88]), where no adversary should be able to forge a valid message-signature pair, even with access to the signing oracle, for a new message.

A one-time signature is defined by the same algorithms $\mathsf{OT.KG}$, $\mathsf{OT.Sign}$, and $\mathsf{OT.Verify}$, but just requires this security level, after at most one signing query.

A.3 SPHF-Friendly Commitment Schemes

In this section, we provide a more formal definition of SPHF-friendly commitment schemes, slightly improving on [ABB+13].

SPHF-Friendly Commitment Schemes. Such an SPHF-friendly commitment is defined by the following algorithms:

– $\mathsf{C.Setup}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the global CRS crs to all other algorithms;
– $\mathsf{C.SetupT}(1^{\mathfrak{K}})$ is an alternative to $\mathsf{C.Setup}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ;
– $\mathsf{C.Com}^{\ell}(\boldsymbol{M})$ takes as input a label ℓ and a message $\boldsymbol{M}$, and outputs a pair (C, δ), where C is the commitment of $\boldsymbol{M}$ for the label ℓ, and δ is the corresponding opening data (a.k.a., decommitment information);
– $\mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta)$ takes as input a commitment C, a label ℓ, a message $\boldsymbol{M}$, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, $\boldsymbol{M}$ and ℓ. It always outputs 0 (false) on $\boldsymbol{M} = \perp$;
– $\mathsf{C.Sim}^{\ell}(\tau)$ takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
– $\mathsf{C.Open}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$ takes as input a commitment C, a label ℓ, a message $\boldsymbol{M}$, and an equivocation key eqk for this commitment, and outputs an opening data δ for C and ℓ on $\boldsymbol{M}$.
– $\mathsf{C.Ext}^{\ell}(\tau, C)$ takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message $\boldsymbol{M}$, or $\perp$ if the commitment is invalid;
– $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M})$ takes as input the trapdoor τ, a commitment C, a message $\boldsymbol{M}$ and a label ℓ, and outputs 0 if the commitment is not perfectly binding to $\boldsymbol{M}$, i.e., if there exists $\boldsymbol{M}' \neq \boldsymbol{M}$ and δ, such that $\mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}', \delta) = 1$.

Correctness. An SPHF-friendly commitment first has to verify the following properties:

– for all correctly generated CRS crs, all commitments and opening data honestly generated pass the verification test: $\forall \ell \forall \boldsymbol{M}, (C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell}(\boldsymbol{M}) \Rightarrow \mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta) = 1$;
– all simulated commitments can be opened on any message:
$$\forall \ell \forall \boldsymbol{M}, ((C, \mathsf{eqk}) \stackrel{\$}{\leftarrow} \mathsf{C.Sim}^{\ell}(\tau) \ \wedge \ \delta \leftarrow \mathsf{C.Open}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})) \Rightarrow \mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta) = 1;$$

- all commitments honestly generated can be correctly extracted:
  $\forall \ell \forall \boldsymbol{M}, (C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell}(\boldsymbol{M}) \Rightarrow \mathsf{C.Ext}^{\ell}(\tau, C) = \boldsymbol{M}$;
- all commitments honestly generated are considered binding by $\mathsf{C.IsBinding}$, with overwhelming probability:
  $\forall \ell \forall \boldsymbol{M}, (C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell}(\boldsymbol{M}) \Rightarrow \mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M}) = 1$;
- all commitments C under some label ℓ for which $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M}) = 1$ are such that for all $\boldsymbol{M'} \neq \boldsymbol{M}$ and δ, $\mathsf{C.Ver}^{\ell}(C, \boldsymbol{M'}, \delta) = 0$.

Of course, to be SPHF-friendly, the commmitment scheme has to admit an SPHF for the following language:

$$\mathcal{L}_{\mathsf{full\text{-}aux}} = \{(\ell, C) \mid \exists \delta,\ \mathsf{C.Ver}^{\ell}(C, \boldsymbol{M}, \delta) = 1\},$$

where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$ and $\boldsymbol{M} = \mathsf{aux}$.

We now list the additional security properties that these algorithms have to satisfy.

Setup Indistinguishability. One should not be able to distinguish the CRS generated by $\mathsf{C.Setup}$ from the one generated by $\mathsf{C.SetupT}$. The commitment scheme is said (t, ε)-setup-indistinguishable if the two distributions for CRSs are (t, ε)-computationally indistinguishable. We denote $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}(t)$ the distance between the two distributions.

Strong Simulation Indistinguishability. Let us denote $\mathsf{C.SCom}$ the algorithm that takes as input the trapdoor τ, a label ℓ and a message $\boldsymbol{M}$ and which outputs $(C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.SCom}^{\ell}(\tau, \boldsymbol{M})$, computed as $(C, \mathsf{eqk}) \stackrel{\$}{\leftarrow} \mathsf{C.Sim}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{C.Open}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$.

One should not be able to distinguish a real commitment (generated by $\mathsf{C.Com}$) from a fake commitment (generated by $\mathsf{C.SCom}$), even with oracle access to the extraction oracle ($\mathsf{C.Ext}$), the binding test oracle ($\mathsf{C.IsBinding}$), and to fake commitments (using $\mathsf{C.SCom}$). The commitment scheme is said (t, ε)-strongly-simulation-indistinguishable if $\mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}(t) \leq \varepsilon$, according to $\mathsf{Exp}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 3.

Remark 1. In this experiment, as in the following ones, the oracle $\mathsf{C.SCom}$ is supposed to store each query/answer $(\ell, \boldsymbol{M}, C)$ in a list Λ and $\mathsf{C.Ext}$-queries on such an $\mathsf{C.SCom}$-output (ℓ, C) are answered by $\boldsymbol{M}$ (as it would be when using $\mathsf{C.Com}$ instead of $\mathsf{C.SCom}$). The same way, $\mathsf{C.IsBinding}$ returns 1 on such commitments (although it is not the case). This is just to exclude trivial attacks.

Robustness. One should not be able to produce a commitment and a label that extracts to $\boldsymbol{M}$ (possibly $\boldsymbol{M} = \perp$) such that $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M}) = 0$, even with oracle access to the extraction oracle ($\mathsf{C.Ext}$), the binding test oracle ($\mathsf{C.IsBinding}$), and to fake commitments (using $\mathsf{C.SCom}$). The commitment scheme is said (t, ε)-robust if $\mathsf{Succ}^{\mathtt{robust}}(t) \leq \varepsilon$, according to the experiment $\mathsf{Exp}^{\mathtt{robust}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 3.

$\mathsf{Exp}^{\mathtt{s\text{-}sim\text{-}ind}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$	$\mathsf{Exp}^{\mathtt{robust}}_{\mathcal{A}}(\mathfrak{K})$
$(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{C.SetupT}(1^{\mathfrak{K}})$;	$(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{C.SetupT}(1^{\mathfrak{K}})$
$(\ell, \boldsymbol{M}, \mathsf{st}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot), \mathsf{C.Ext}^{\cdot}(\tau,\cdot), \mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{crs})$	$(C, \ell) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot), \mathsf{C.Ext}^{\cdot}(\tau,\cdot), \mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{crs})$
if $b = 0$ **then** $(C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell}(\boldsymbol{M})$	$\boldsymbol{M} \leftarrow \mathsf{C.Ext}^{\ell}(\tau, C)$
else $(C, \delta) \stackrel{\$}{\leftarrow} \mathsf{C.SCom}^{\ell}(\tau, \boldsymbol{M})$	**if** $(\ell, \boldsymbol{M}, C) \in \Lambda$ **then return** 0
return $\mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot), \mathsf{C.Ext}^{\cdot}(\tau,\cdot), \mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{st}, C, \delta)$	**if** $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M}) = 1$ **then return** 0
	return 1

Fig. 3. Strong Simulation Indistinguishability and Strong Binding Extractability (Λ is defined in Remark 1)

Pseudo-Randomness vs. Strong Pseudo-Randomness. As in [ABB+13], from the smoothness of the SPHF on $\mathcal{L}_{(\mathsf{crs},\mathsf{aux})}$, one can derive the *pseudo-randomness* property on SPHF-friendly commitments, modeled by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-ps-rand}}$ in Figure 4.

If the adversary $\mathcal{A}$ is given a commitment C by $\mathsf{C.Sim}$ with label ℓ (adversary-chosen), even with access to the oracles $\mathsf{C.SCom}$, $\mathsf{C.Ext}$, and $\mathsf{C.IsBinding}$, then for any $\boldsymbol{M}$, it cannot distinguish the hash value of (ℓ, C) on language $\mathcal{L}_{(\mathsf{crs},\boldsymbol{M})}$ from a random value, while being given hp, since C could have been generated as $\mathsf{C.Com}^{\ell}(\boldsymbol{M}'')$ for some $\boldsymbol{M}'' \neq \boldsymbol{M}$, which excludes it to belong to $\mathcal{L}_{(\mathsf{crs},\boldsymbol{M})}$, under the robustness. In the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-ps-rand}}$, we let the adversary choose $(\ell, \boldsymbol{M})$, and we have: $\mathsf{Adv}^{\texttt{c-ps-rand}}(t) \leq \mathsf{Adv}^{\texttt{s-sim-ind}}(t) + \mathsf{Succ}^{\texttt{robust}}(t) + \mathsf{Adv}^{\texttt{smooth}}$.

Note that when hk and hp do not depend on $\boldsymbol{M}$ nor on C, and when the smoothness holds even if the adversary can choose C after having seen hp (i.e., the SPHF is actually a KV-SPHF [BBC+13b]), they can be generated from the beginning of the game, with hp given to the adversary much earlier.

$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-ps-rand-}b}(\mathfrak{K})$	$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-}b}(\mathfrak{K})$
$(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{C.SetupT}(1^{\mathfrak{K}})$	$(\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{C.SetupT}(1^{\mathfrak{K}})$
$(\ell, \boldsymbol{M}, \mathsf{st}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{crs})$	$(\ell, \boldsymbol{M}, \mathsf{st}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{crs})$
$(C, \mathsf{eqk}) \stackrel{\$}{\leftarrow} \mathsf{C.Sim}^{\ell}(\tau)$	$(C, \mathsf{eqk}) \stackrel{\$}{\leftarrow} \mathsf{C.Sim}^{\ell}(\tau)$
$\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$	$\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$
$\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, (\mathsf{crs}, \boldsymbol{M}), (\ell, C))$	$\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, (\mathsf{crs}, \boldsymbol{M}), \perp)$
if $b = 0$ **then**	**if** $b = 0$ **then**
$\quad H \leftarrow \mathsf{Hash}(\mathsf{hk}, (\mathsf{crs}, \boldsymbol{M}), (\ell, C))$	$\quad H \leftarrow \mathsf{Hash}(\mathsf{hk}, (\mathsf{crs}, \boldsymbol{M}), (\ell, C))$
else	**else**
$\quad H \stackrel{\$}{\leftarrow} \Pi$	$\quad H \stackrel{\$}{\leftarrow} \Pi$
$b \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{st}, C, \mathsf{hp}, H)$	$(\ell', C', \mathsf{st}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{st}, C, \mathsf{hp}, H)$
return b	**if** $(\ell', ?, C') \in \Lambda$ **then**
	$\quad H' \leftarrow \perp$
	else
	$\quad H' \leftarrow \mathsf{Hash}(\mathsf{hk}, (\mathsf{crs}, \boldsymbol{M}), (\ell', C'))$
	return $\mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{st}, H')$

Fig. 4. Pseudo-Randomness and Strong Pseudo-Randomness (Λ is defined in Remark 1)

However, for our PAKE protocols, as for those in [ABB+13], one needs a stronger property called *strong pseudo-randomness*. It is modelled by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand}}$ depicted in Figure 4. This property is only defined for SPHF-friendly commitment with a KV-SPHF.

It is similar to the pseudo-randomness game except the adversary can also ask a hash value of a commitment C' under a label ℓ' (under the restriction that (ℓ', C') was not generated by $\mathsf{C.SCom}$) under the hashing key hk.

Generically, a property like the 2-universality of [CS02] may be needed for the SPHF. However, for our new commitment scheme and the one in [ABB+13], this property holds directly, while the used SPHF is not 2-universal (and so may be more efficient).

A.4 Ideal Functionalities

UC-Secure Oblivious Transfer. The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Figure 5. It is inspired from [CKWZ13].

UC-Secure Password-Authenticated Key Exchange. We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 6). It was described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties P_1,…,P_n via the following queries:

- **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ from party P_i**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ from party P_j**, with $s \in \{1, \ldots, k\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Receive-message with the same ssid from P_j.
- **Upon receiving a message $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.
- **Upon receiving a message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m_s)$ to P_j and ignore further Received-message with the same ssid from the adversary.

Fig. 5. Ideal Functionality for 1-out-of-k Oblivious Transfer $\boldsymbol{\mathcal{F}_{(1,k)\text{-OT}}}$

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties P_1,…,P_n via the following queries:

- **Upon receiving a query $(\mathtt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \pi)$ from party P_i:**
  Send $(\mathtt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to $\mathcal{S}$. If this is the first NewSession query, or if this is the second NewSession query and there is a record $(\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \pi')$, then record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \pi)$ and mark this record fresh.
- **Upon receiving a query $(\mathtt{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, \pi')$ from the adversary S:**
  If there is a record of the form (P_i, P_j, π) which is fresh, then do: If $\pi = \pi'$, mark the record compromised and reply to $\mathcal{S}$ with "correct guess". If $\pi \neq \pi'$, mark the record interrupted and reply with "wrong guess".
- **Upon receiving a query $(\mathtt{NewKey}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{SK})$ from the adversary S:**
  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \pi)$, and this is the first NewKey query for P_i, then:
  - If this record is compromised, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{SK})$ to player P_i.
  - If this record is fresh, and there is a record (P_j, P_i, π') with $\pi' = \pi$, and a key SK' was sent to P_j, and (P_j, P_i, π) was fresh at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{SK}')$ to P_i.
  - In any other case, pick a new random key SK' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \pi)$ as completed.

Fig. 6. Ideal Functionality for PAKE $\boldsymbol{\mathcal{F}_{\text{pwKE}}}$

as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the `NewKey`-query, it additionally learns the session key.

B Semi-Adaptive OT and PAKE

In this appendix, we provide the complete proofs for the security of the OT and PAKE protocols from Section 4: security holds in the UC-framework, against semi-adaptive adversaries, without requiring reliable erasures.

B.1 Proof Security of our OT Scheme

To prove the security of our OT protocol (see Section 4.2), in the UC-framework, against semi-adaptive adversaries but without erasures, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$. Essentially, we do the following:

1. we make the setup algorithm additionally output the trapdoor (setup-indistinguishability);
2. we then replace all the commitment queries by simulated (fake) commitments (simulation-indistinguishability);
3. when simulating a sub-session between two honest receivers, the simulator commits to an arbitrary value s (e.g., $s = 0$ — hiding property of the commitment) and uses arbitrary messages (e.g., $m_t = 0$ for all t — pseudo-randomness of the SPHF on robust commitment).

Recall that no corruption is authorized in this case. We now just need to deal with sub-session where either the sender or the receiver is corrupted:

4. when simulating a honest receiver, when the (corrupted) sender submits the values $(\mathsf{hp}_t, M_t)_t$ and the simulator can extract all the messages thanks to the trapdoor (simulatability of the commitment). This allows to simulate the `Send`-query to the ideal functionality;
5. when simulating a honest sender, the simulator extracts the committed value s from the commitment C of the (corrupted) receiver. This allows to simulate the `Receive`-query to the ideal functionality, which gives back the message m_s that should be received. We can then use this value m_s instead of the one provided by the environment.
6. still when simulating a honest sender, the simulator simulate (SimKG) projection keys and use random messages m_t for $t \neq s$ (smoothness of the SPHF on robust commitment and GL-indistinguishability of the EPHF). In case of corruption, we get the correct messages m_t for $t \neq s$ (m_s being already known), that we can explain using $\mathsf{Explain}$.

On one hand, if the adversary corrupts a sender with input s and plays honestly the protocol with s, the simulator will extract correctly s from the commitment C of the sender (trapdoor correctness of extractable commitments) and do a `Send`-query with the same s. On the other hand, if the adversary corrupts a receiver with inputs $(M_t)_t$ and plays honestly the protocol with $(M_t)_t$, the simulator will compute correctly the hash values H_t and so extract correctly $(M_t)_t$ and do a `Receive`-query with the same messages $(M_t)_t$. This means that in both cases, the extracted value is the one used by the adversary, which property is called *input-preserving*, as required by the definition of semi-adaptivity of Garay *et al.* [GWZ09]. Another property required by their definition is the so-called *setup-adaptive simulation*, which says that the CRS or any setup is generated independent of which party will be corrupted. This is clearly the case of our simulation. So the protocol is semi-adaptive.

Let us now go into more details:

Game $\mathbf{G}_0$: This is the **real game**.

Game $\mathbf{G}_1$: In this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the inputs $(m_1, \ldots, m_k)$ and s sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\mathbf{G}_2$: In this game, we just replace the setup algorithm $\mathsf{C.Setup}$ by $\mathsf{C.SetupT}$ that additionally outputs the trapdoor $(\mathsf{crs}, \tau) \xleftarrow{\$} \mathsf{C.SetupT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under setup indistinguishability. Corruptions are handled the same way.

Game $\mathbf{G}_3$: We first deal with **honest receivers** P_j: we replace all the commitments $(C, \delta) \xleftarrow{\$} \mathsf{C.Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ in Step 1 of the index query phase of honest receivers by simulated commitments $(C, \delta) \xleftarrow{\$} \mathsf{C.SCom}^{\ell}(\tau, s)$, which means $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{C.Sim}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{C.Open}^{\ell}(\mathsf{eqk}, C, s)$. We then store (ℓ, s, C, δ) in Λ.

With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{sim-ind}}$ security game for each session, in which $\mathsf{C.SCom}$ is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games. In case of corruption of the receiver, we just learn the already known value s.

Game $\mathbf{G}_4$: We now deal with sub-sessions between an **honest sender** P_i and an **honest receiver** P_j: on behalf of the receiver, the simulator computes K_t as the sender does, i.e., using hk_t instead of hp_t and δ (for all t): $K_t = \mathsf{Hash}(\mathsf{hk}_t, (\mathsf{crs}, t), (\ell, C))$. Notice that hk_t is known since it has also been generated by the simulator on behalf of the sender.
This game is indistinguishable from the previous one, thanks to the correctness of the SPHF.

Game $\mathbf{G}_5$: Still in this case, we replace K_t by a random value (both for the sender and the receiver and for all t). This game is indistinguishable from the previous one, thanks to the pseudo-randomness of the SPHF. The basic pseudo-randomness game can be used since the sender and the receiver cannot be corrupted in this case (because the adversary is semi-adaptive) and so, neither C nor any hk_t never have to be revealed later.

Game $\mathbf{G}_6$: Still in this case, instead of using the messages $(m_1, \ldots, m_t)$ provided by the environment, we use $(m'_1, \ldots, m'_k) = (0, \ldots, 0)$. Since the masks K_t (for all t) are random, this game is perfectly indistinguishable form the previous one.
When simulating sub-sessions between two honest players, we do not use the inputs provided by the environment (except s to compute the opening information δ, but δ actually is not used, and so s could be chosen arbitrarily).
From now on, we only consider sub-sessions where at least one player is corrupted.

Game $\mathbf{G}_7$: We now deal with **honest senders** P_i: when receiving a commitment C (from a corrupted receiver P_j), the simulator extracts the committed value s and aborts if it is not binding, i.e., if $\mathsf{C.IsBinding}^\ell(\tau, C, s) = 0$.
With an hybrid proof, applying the robustness property of the commitment scheme, for every honest sender, this game is indistinguishable from the previous one, since it is hard for an adversary to generate non-binding commitments. Notice that labels are important here and enables the simulator to extract C (for label ℓ) and call $\mathsf{C.IsBinding}$ on it, because even if C is replayed, it cannot be replayed with the same label.

Game $\mathbf{G}_8$: Still in this case, when receiving a commitment C (from a corrupted receiver P_j), the simulator extracts the committed value s. For $t \neq s$, instead of generating hk_t and hp_t honestly using HashKG and ProjKG, it generates $(\mathsf{hp}, \mathsf{expk}) \overset{\$}{\leftarrow} \mathsf{SimKG}(\mathsf{crs}, \tau, C)$, chooses $K_t \overset{\$}{\leftarrow} \Pi = \{0,1\}^{\nu_m}$ and sets $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, (\mathsf{crs}, t), (\ell, C), K_t, \mathsf{expk})$.
With an hybrid proof, applying the GL-indistinguishability of EPHF for every honest sender, on every index $t \neq s$, since C is not in $\mathcal{L}_{(\mathsf{crs},t)}$ because $\mathsf{C.IsBinding}^\ell(\tau, C, s) = 1$, this game is indistinguishable from the previous one.

Game $\mathbf{G}_9$: We do not use anymore the knowledge of $(m_t)_{t \neq s}$ when simulating an **honest sender** P_i: when receiving a commitment C, the simulator extracts the committed value s. For all $t \neq s$, instead of choosing a random K_t, and setting $\mathsf{hk}_t \overset{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, (\mathsf{crs}, t), (\ell, C), H, \mathsf{expk})$ and $M_t \leftarrow K_t \operatorname{xor} m_t$, the simulator chooses a random $M_t \in \Pi$, and does not generate hk_t, since it has to be given to the adversary only in case of corruption. In case of corruption, the simulator learns m_t and can set $K_t \leftarrow M_t \operatorname{xor} m_t$, and $\mathsf{hk}_t \overset{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}, (\mathsf{crs}, t), (\ell, C), H, \mathsf{expk})$.
The distributions of M_t and K_t are left unchanged since K_t was random (and m_t was independent of K_t). Therefore, this game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_{10}$: We do not use anymore the knowledge of s when simulating an **honest receiver** P_j: the simulator generates $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{C.Sim}^\ell(\tau)$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$, to send C during the index query phase of honest receivers. It then stores $(\ell, \perp, C, \mathsf{eqk})$ in Λ. We essentially break the atomic $\mathsf{C.SCom}$ in the two separated processes $\mathsf{C.Sim}$ and $\mathsf{C.Open}$. This does not change anything from the previous game since δ is never revealed. Λ is first filled with $(\ell, \perp, C, \mathsf{eqk})$, it can be updated with correct values in case of corruption of the receiver.
When it thereafter receives $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (\mathsf{hp}_1, M_1, \ldots, \mathsf{hp}_k, M_k))$ from the adversary, the simulator computes, for $t = 1, \ldots, k$, opening values $\delta_t \leftarrow \mathsf{C.Open}^\ell(\mathsf{eqk}, C, t)$, the masks $K_t \leftarrow$

$\mathsf{ProjHash}(\mathsf{hp}_t, (\mathsf{crs}, t), (\ell, C), \delta_t)$ and $m_t = K_t \operatorname{xor} M_t$. This provides the database submitted by the sender.

Game $\mathbf{G}_{11}$: We can now make use of the ideal functionality, without knowing the inputs from the environment.

B.2 Proof Security of our PAKE Scheme

To prove the security of our PAKE protocol (see Section 4.3), in the UC-framework, against semi-adaptive adversaries but without erasures, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

We say that a flow is *oracle-generated* if the pair (hp, C) was sent by an honest player (or the simulator) and received without any alteration by the expected receiver. It is said *non-oracle-generated* otherwise.

The steps in the proof are similar to the previous proof. Here are detailed games:

Game $\mathbf{G}_0$: This is the **real game.**

Game $\mathbf{G}_1$: First, in this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the inputs π_i and π_j sent by the environment to the players. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\mathbf{G}_2$: We now replace the setup algorithm $\mathsf{C.Setup}$ by $\mathsf{C.SetupT}$ that additionally outputs the trapdoor $(\mathsf{crs}, \tau) \xleftarrow{\$} \mathsf{C.SetupT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under setup indistinguishability. Corruptions are handled the same way.

Game $\mathbf{G}_3$: We now deal with **honest players** P_i receiving an oracle-generated flow (hp_j, C_j) from P_j, with **a different password**: $\pi_j \neq \pi_i$. In this case, P_i and P_j are honest at the beginning and so a semi-adaptive adversary cannot corrupt any of them. So, we can replace the hash value $H_i = \mathsf{Hash}(\mathsf{hk}_i, (\mathsf{crs}, \pi_i), (\ell_j, C_j))$ by a random value. This game is indistinguishable from the previous one thanks to the smoothness of the SPHF.

Game $\mathbf{G}_4$: Still in this case, we replace $\mathsf{SK}_i = H_i \operatorname{xor} H'_j$ by a random value. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_5$: We now deal with all honest players. We replace all the commitments $(C, \delta) \xleftarrow{\$} \mathsf{C.Com}^{\ell}(\pi)$ ($\pi = \pi_i$ or π_j) with $\ell = \ell_i$ or ℓ_j by simulated commitments $(C, \delta) \xleftarrow{\$} \mathsf{C.SCom}^{\ell}(\tau, \pi)$, which means $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{C.Sim}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{C.Open}^{\ell}(\mathsf{eqk}, C, s)$. We then store (ℓ, π, C, δ) in Λ.
With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{sim-ind}}$ security game for each session, in which $\mathsf{C.SCom}$ is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games.

Game $\mathbf{G}_6$: We now deal with **honest players** P_i receiving an oracle-generated flow (hp_j, C_j) from P_j, with the **same password** as P_i: $\pi_j = \pi_i$. We remark that the hash value $H'_i = \mathsf{ProjHash}(\mathsf{hp}_j, (\mathsf{crs}, \pi_i), (\ell_i, C_i), \delta_i)$ computed by the player P_i using δ_i is equal to the hash value $H_i = \mathsf{Hash}(\mathsf{hk}_j, (\mathsf{crs}, \pi_j), (\ell_i, C_i))$ that P_j would compute if he gets the oracle-generated generated flow (hp_i, C_i) sent by P_i. So the first time we need to compute one of this values (H_i or H'_i), we compute it as $\mathsf{Hash}(\mathsf{hk}_j, (\mathsf{crs}, \pi_j), (\ell_i, C_i))$, and if the other value needs to be computed, we just sets it equal to the first one.
Therefore, in this case, δ_i is no more used, since a semi-adaptive adversary is not allowed to corrupt P_i.
This game is indistinguishable from the previous one due to the correctness of the SPHF.

Game $\mathbf{G}_7$: Still in this case, we replace H'_i (and H_i if P_j received the oracle-generated flow generated flow sent by P_i) by a random value.
To prove this game is indistinguishable from the previous one, we consider two cases:

- P_j received the oracle-generated flow generated by P_i. In this case, hk_j is only used to compute $H_i = H'_i$, and since δ_i is no more used, we can apply the pseudo-randomness game on C_i to prove that $H_i = H'_i$ is indistinguishable from random;
- P_j received a non-oracle-generated flow (hp'_i, C'_i). In this case hk_j is only used to compute $H'_i = \mathsf{Hash}(\mathsf{hk}_j, (\mathsf{crs}, \pi_i), (\ell_i, C_i))$ and $H_i = \mathsf{Hash}(\mathsf{hk}_j, (\mathsf{crs}, \pi_i), (\ell_i, C'_i))$. In this case, we can apply the strong pseudo-randomness game to prove that H'_i still looks random.

Game $\mathbf{G}_8$: Still in this case, we replace $\mathsf{SK}_i \leftarrow H'_i \operatorname{xor} H_j$ by a random value, and if P_j also received the oracle-generated flow sent by P_i, then we set $\mathsf{SK}_j = \mathsf{SK}_i$. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_9$: In this game, if P_i receives a non-oracle-generated flow (hp_j, C_j), then we extract π_j from C_j. If $\pi_j \neq \pi_i$, then we check if $\mathsf{C.IsBinding}^{\ell_i}(\tau, C_j, \pi_j) = 1$ and aborts if this is not the case. With an hybrid proof, applying the robustness property, for every P_i, this game is indistinguishable from the previous one.

Game $\mathbf{G}_{10}$: In this game, we deal with the case when P_i receives a non-oracle-generated flow such that the extracted π_j is not equal to π_i. In this case, instead of generating hk_i and hp_i honestly using HashKG and ProjKG, we generate $(\mathsf{hp}_i, \mathsf{expk}_i) \stackrel{\$}{\leftarrow} \mathsf{SimKG}(\mathsf{crs}, \tau, \perp)$ and choose $H_j \stackrel{\$}{\leftarrow} \mathit{\Pi}$ and sets $\mathsf{hk}_i \stackrel{\$}{\leftarrow} \mathsf{Explain}(\mathsf{hp}_i, (\mathsf{crs}, \pi_i), (\ell_j, C_j), H_j, \mathsf{expk}_i)$. With an hybrid proof, applying the KV-indistinguishability property of EPHF, for every P_i, this game is indistinguishable from the previous one.

Game $\mathbf{G}_{11}$: Still in the same case, we choose SK_i at random and sets $H_j \leftarrow H'_i \operatorname{xor} \mathsf{SK}_i$. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_{12}$: We can now make use of the ideal functionality, without knowing the inputs from the environment.

C New SPHF-Friendly Commitment Scheme

In this appendix, we completely describe our new commitment scheme, with the description of the associated SPHF. Security proofs are also provided.

C.1 The Commitment Scheme

We start by a complete description of the commitment scheme:

- $\mathsf{C.SetupT}(1^{\mathfrak{K}})$ generates a cyclic group $\mathbb{G}$ of order p, together with three generators g, h, $\hat{h}$, a tuple $(\alpha, \beta, \gamma, \alpha', \beta', \gamma') \leftarrow \mathbb{Z}_p^6$, and H is a random collision-resistant hash function from some family $\mathcal{H}$. It then computes the tuple $(c = g^\alpha \hat{h}^\gamma, d = g^\beta h^\gamma, c' = g^{\alpha'} \hat{h}^{\gamma'}, d' = g^{\beta'} h^{\gamma'})$. It also generates a random scalar $t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sets $T = g^t$. The CRS crs is set as $(g, h, \hat{h}, H, c, d, c', d', T)$ and the trapdoor τ is the decryption key $(\alpha, \alpha', \beta, \beta', \gamma, \gamma')$ (a.k.a., extraction trapdoor) together with t (a.k.a., equivocation trapdoor).
  For $\mathsf{C.Setup}(1^{\mathfrak{K}})$, the CRS is generated the same way, but forgetting the scalars, and thus without any trapdoor;
- $\mathsf{C.Com}^\ell(\boldsymbol{M})$, for $\boldsymbol{M} = (M_i)_i \in \{0,1\}^m$ and a label ℓ, works as follows: For $i = 1, \dots, m$, it chooses two random scalars $r_{i,M_i}, s_{i,M_i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and set:
  $$\begin{array}{llll} e_{i,M_i} = g^{r_{i,M_i}} & u_{i,M_i} = g^{s_{i,M_i}} & v_{i,M_i} = \hat{h}^{r_{i,M_i}} h^{s_{i,M_i}} & w_{i,M_i} = (c^{r_{i,M_i}} \cdot d^{s_{i,M_i}}) \cdot (c'^{r_{i,M_i}} d'^{s_{i,M_i}})^\xi \\ e_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{G} & u_{i,\overline{M}_i} = T / u_{i,M_i} & v_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{G} & w_{i,\overline{M}_i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \end{array}$$
  with $\xi = H(\ell, (e_{i,b}, u_{i,b}, v_{i,b})_{i,b})$. The commitment is $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b} \in \mathbb{G}^{8m}$, while the opening information is the $2m$-tuple $\delta = (r_{i,M_i}, s_{i,M_i})_i \in \mathbb{Z}_p^{2m}$.
- $\mathsf{C.Ver}^\ell(C, \boldsymbol{M}, \delta)$ just checks all the above equalities ($=$);

– $\mathsf{C.Sim}^\ell(\tau)$ takes as input the equivocation trapdoor, namely the scalar t, and outputs the tuple $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b}$ and $\mathsf{eqk} = ((r_{i,b})_i, (s_{i,b})_{i,b})$, where, for $i = 1, \ldots, m$, $r_{i,0}, r_{i,1}, s_{i,0} \xleftarrow{\$} \mathbb{Z}_p$, $s_{i,1} = t - s_{i,0}$:

$$\begin{array}{llll} e_{i,0} = g^{r_{i,0}} & u_{i,0} = g^{s_{i,0}} & v_{i,0} = \hat{h}^{r_{i,0}} h^{s_{i,0}} & w_{i,0} = (c^{r_{i,0}} \cdot d^{s_{i,0}}) \cdot (c'^{r_{i,0}} \cdot d'^{s_{i,0}})^\xi \\ e_{i,1} = g^{r_{i,1}} & u_{i,1} = g^{s_{i,1}} = T/u_{i,0,1} & v_{i,1} = \hat{h}^{r_{i,0}} h^{s_{i,1}} & w_{i,1} = (c^{r_{i,1}} \cdot d^{s_{i,1}}) \cdot (c'^{r_{i,1}} \cdot d'^{s_{i,1}})^\xi; \end{array}$$

– $\mathsf{C.Open}^\ell(\mathsf{eqk}, C, \boldsymbol{M})$ simply extracts the useful values from $\mathsf{eqk} = \boldsymbol{s}$ to make the opening value $\delta = (r_{i,M_i}, s_{i,M_i})_i$ in order to open to $\boldsymbol{M} = (M_i)_i$.
– $\mathsf{C.Ext}^\ell(\tau, C)$ outputs $\perp$ if $u_{i,0} \cdot u_{i,1} \neq T$. It also outputs $\perp$ if for some i, for both $b = 0$ and $b = 1$ or for none of them:
$$w_{i,b} = e_{i,b}^{\alpha+\xi\alpha'} \cdot u_{i,b}^{\beta+\xi\beta'} \cdot v_{i,b}^{\gamma+\xi\gamma'}.$$
Otherwise, for each i, there is exactly one bit b verifying the above equality; and it sets M_i to this bit b, and returns the resulting message $\boldsymbol{M} = (M_i)_i$.
– $\mathsf{C.IsBinding}^\ell(\tau, C, \boldsymbol{M})$ outputs 1 if and only if

$$\begin{cases} w_{i,\overline{M}_i} \neq e_{i,\overline{M}_i}^{\alpha+\xi\alpha'} \cdot u_{i,\overline{M}_i}^{\beta+\xi\beta'} \cdot v_{i,\overline{M}_i}^{\gamma+\xi\gamma'} & \text{for all } i = 1, \ldots, m, \text{ if } \boldsymbol{M} \neq \perp \\ w_{i,b} \neq e_{i,b}^{\alpha+\xi\alpha'} \cdot u_{i,b}^{\beta+\xi\beta'} \cdot v_{i,b}^{\gamma+\xi\gamma'} & \text{for some } i, \text{ for } b = 0, 1, \text{ if } \boldsymbol{M} = \perp \end{cases}$$

Since the requirement on $\mathsf{C.IsBinding}$ is just to accept honestly generated commitments but to reject a commitment with any message $\boldsymbol{M}$ if the verification algorithm could accept another message $\boldsymbol{M'}$, several definitions could be acceptable. But the above one is enough for our purpose. Then, correctness and setup indistinguishability are straightforward to prove. Strong simulation indistinguishability, robustness and strong pseudo-randomness are proven in the next sections. But before that, let us just give the detailed construction of the associated SPHFs and a variant of this first scheme, called the second scheme.

C.2 SPHF

To construct SPHFs for our commitment schemes, we use the framework from [BBC+13b].

KV-SPHF. Let us consider a commitment $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b}$ under some label ℓ. From the definition, this is a commitment of $\boldsymbol{M}$ if, for $i = 1, \ldots, m$, $(e_{i,M_i}, e_{i,M_i}^\xi, u_{i,M_i}, u_{i,M_i}^\xi, v_{i,M_i}, w_{i,M_i})$ is a linear combination of the rows, with coefficients $(r_{i,M_i}, r_{i,M_i}\xi, s_{i,M_i}, s_{i,M_i}\xi)$, of the following matrix:

$$\Gamma = \begin{pmatrix} g & 1 & 1 & 1 & \hat{h} & c \\ 1 & g & 1 & 1 & 1 & c' \\ 1 & 1 & g & 1 & h & d \\ 1 & 1 & 1 & g & 1 & d' \end{pmatrix}. \qquad (1)$$

Therefore, the hashing key is a random tuple $\mathsf{hk} = (\eta_{i,j})_{i,j} \xleftarrow{\$} \mathbb{Z}_p^{m\times 6}$, the projection key is $\mathsf{hp} = (\mathsf{hp}_{i,k})_{i,k} \in \mathbb{G}^{m\times 4}$ with:

$$\begin{aligned} \mathsf{hp}_{i,1} &= g^{\eta_{i,1}} \cdot \hat{h}^{\eta_{i,5}} \cdot c^{\eta_{i,6}} \\ \mathsf{hp}_{i,2} &= g^{\eta_{i,2}} \cdot c'^{\eta_{i,6}} \\ \mathsf{hp}_{i,3} &= g^{\eta_{i,3}} \cdot h^{\eta_{i,5}} \cdot c^{\eta_{i,6}} \\ \mathsf{hp}_{i,4} &= g^{\eta_{i,4}} \cdot d'^{\eta_{i,6}} \end{aligned}$$

and the hash value is:

$$\begin{aligned}\mathsf{Hash}(\mathsf{hk},(\mathsf{crs},\boldsymbol{M}),(\ell,C)) &\coloneqq \prod_{i=1}^{m}\left(e_{i,M_i}^{\eta_{i,1}+\xi\eta_{i,2}}\cdot u_{i,M_i}^{\eta_{i,3}+\xi\eta_{i,4}}\cdot v_{i,M_i}^{\eta_{i,5}}\cdot w_{i,M_i}^{\eta_{i,6}}\right)\\ &= \prod_{i=1}^{m}\left(\mathsf{hp}_{i,1}^{r_{i,M_i}}\cdot \mathsf{hp}_{i,2}^{\xi r_{i,M_i}}\cdot \mathsf{hp}_{i,3}^{s_{i,M_i}}\cdot \mathsf{hp}_{i,4}^{s_{i,M_i}}\right)\\ &=: \mathsf{ProjHash}(\mathsf{hp},(\mathsf{crs},\boldsymbol{M}),(\ell,C),\delta).\end{aligned}$$

GL-SPHF. For a GL-SPHF, ξ is known in advance. So we can use a simpler matrix Γ. In addition, we can re-use the same hp for all bits of the commitment by using a scalar ε of at least $\mathfrak{K}$ bits (for the sake of simplicity, we suppose that $\varepsilon \xleftarrow{\$} \mathbb{Z}_p$). Also notice that ε is not required when $m = 1$.

More precisely, C is a commitment of $\boldsymbol{M}$, if, for $i = 1,\ldots,m$, $(e_{i,M_i}, u_{i,M_i}, v_{i,M_i}, w_{i,M_i})$ is a linear combination of the rows, with coefficients (r_{i,M_i}, s_{i,M_i}), of the following matrix:

$$\Gamma = \begin{pmatrix} g & 1 & \hat{h} & c\cdot c'^{\xi}\\ 1 & g & h & d\cdot d'^{\xi}\end{pmatrix}$$

Therefore the hashing key is a random tuple $\mathsf{hk} = (\eta_1,\eta_2,\eta_3,\eta_4,\varepsilon) \xleftarrow{\$} \mathbb{Z}_p^5$, the projection key is $\mathsf{hp} = (\mathsf{hp}_1,\mathsf{hp}_2,\varepsilon) \in \mathbb{G}^2 \times \mathbb{Z}_p$ with:

$$\begin{aligned}\mathsf{hp}_1 &= g^{\eta_1}\cdot \hat{h}^{\eta_3}\cdot (c\cdot c'^{\xi})^{\eta_4}\\ \mathsf{hp}_2 &= g^{\eta_2}\cdot h^{\eta_3}\cdot (d\cdot d'^{\xi})^{\eta_4}\end{aligned}$$

and the hash value is:

$$\begin{aligned}\mathsf{Hash}(\mathsf{hk},(\mathsf{crs},\boldsymbol{M}),(\ell,C)) &\coloneqq \prod_{i=1}^{m}\left(e_{i,M_i}^{\eta_1}\cdot u_{i,M_i}^{\eta_2}\cdot v_{i,M_i}^{\eta_3}\cdot w_{i,M_i}^{\eta_4}\right)^{\varepsilon^i}\\ &= \prod_{i=1}^{m}\left(\mathsf{hp}_1^{r_{i,M_i}}\cdot \mathsf{hp}_2^{s_{i,M_i}}\right)^{\varepsilon^i}\\ &=: \mathsf{ProjHash}(\mathsf{hp},(\mathsf{crs},\boldsymbol{M}),(\ell,C),\delta).\end{aligned}$$

C.3 Preliminaries: $\boldsymbol{w}$-Pseudo-Randomness

To prove that our commitment scheme is SPHF-friendly, we will first prove an intermediate property we call w-pseudo-randomness, which is defined by the experiments $\mathsf{Exp}^{w\text{-}\mathtt{ps\text{-}rand}\text{-}b}$ in Figure 7, where the Remark 1 from the definitions in Appendix A.3 still applies. It roughly says that simulating commitments with valid $w_{i,\overline{M}_i}$ is indistinguishable from generating them with random $w_{i,\overline{M}_i}$. This can be seen as a pseudo-randomness property for the implicit underlying 2-universal hash proof system.

The proof is close (though slightly different) from the proof of vector-indistinguishability with partial opening under chosen-ciphertexts attacks in [ABB+13]. More precisely, the proof first consists in aborting as soon as the value ξ of a commitment queried to C.Ext or C.IsBinding is equal to the value ξ in our experiment. Thanks to the collision resistance of the hashing function, this is computationally indistinguishable. Then it consists in the following sequence of hybrid games: in the hybrid game i ($i = 0,\ldots,m$),

$$\begin{cases} w_{j,\overline{M}_j} \xleftarrow{\$} \mathbb{G} & \text{for } j = 1,\ldots,i\\ w_{j,\overline{M}_j} \leftarrow (c^{r_{j,\overline{M}_j}}\cdot d^{s_{j,\overline{M}_j}})\cdot (c'^{r_{j,\overline{M}_j}}\cdot d'^{s_{j,\overline{M}_j}})^{\xi} & \text{for } j = i+1,\ldots,m\end{cases}$$

so that the hybrid game 0 is $\mathsf{Exp}_{\mathcal{A}}^{w\text{-}\mathtt{ps\text{-}rand}\text{-}0}$ while the hybrid game m is $\mathsf{Exp}_{\mathcal{A}}^{w\text{-}\mathtt{ps\text{-}rand}\text{-}1}$. It remains to prove that the hybrid game k is indistinguishable from the hybrid game $k+1$. This is done by the following sequence of subgames:

$$
\begin{array}{l}
\mathsf{Exp}_{\mathcal{A}}^{w\text{-}\mathtt{ps}\text{-}\mathtt{rand}\text{-}b}(\mathfrak{K}) \\
\quad (\mathsf{crs}, \tau) \stackrel{\$}{\leftarrow} \mathsf{C.SetupT}(1^{\mathfrak{K}}) \\
\quad (\ell, \boldsymbol{M}, \mathsf{st}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{crs}) \\
\quad \textbf{for } i = 1, \ldots, m \textbf{ do} \\
\qquad r_{i,0}, r_{i,1}, s_{i,0} \stackrel{\$}{\leftarrow} \mathbb{Z}_p; s_{i,1} \leftarrow t - s_{i,0} \\
\qquad e_{i,0} \leftarrow g^{r_{i,0}}; u_{i,0} \leftarrow g^{s_{i,0}}; v_{i,0} = \hat{h}^{r_{i,0}} h^{s_{i,0}} \\
\qquad e_{i,1} \leftarrow g^{r_{i,1}}; u_{i,1} \leftarrow g^{s_{i,1}}; v_{i,1} = \hat{h}^{r_{i,1}} h^{s_{i,1}} \\
\quad \xi = H(\ell, (e_{i,b}, u_{i,b}, v_{i,b})_{i,b}) \\
\quad \textbf{for } i = 1, \ldots, m \textbf{ do} \\
\qquad w_{i,M_i} \leftarrow (c^{r_{i,M_i}} \cdot d^{s_{i,M_i}}) \cdot (c'^{r_{i,M_i}} \cdot d'^{s_{i,M_i}})^{\xi} \\
\qquad \textbf{if } b = 0 \textbf{ then} \\
\qquad\quad w_{i,\overline{M_i}} \leftarrow (c^{r_{i,\overline{M_i}}} \cdot d^{s_{i,\overline{M_i}}}) \cdot (c'^{r_{i,\overline{M_i}}} \cdot d'^{s_{i,\overline{M_i}}})^{\xi} \\
\qquad \textbf{else} \\
\qquad\quad w_{i,\overline{M_i}} \stackrel{\$}{\leftarrow} \mathbb{G} \\
\quad C \leftarrow (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b} \\
\quad \delta \leftarrow (r_{i,M_i}, s_{i,M_i})_i \\
\quad \textbf{return } \mathcal{A}^{\mathsf{C.SCom}^{\cdot}(\tau,\cdot),\mathsf{C.Ext}^{\cdot}(\tau,\cdot),\mathsf{C.IsBinding}^{\cdot}(\tau,\cdot,\cdot)}(\mathsf{st}, C, \delta)
\end{array}
$$

Fig. 7. w-pseudo-randomness

Game $\mathbf{G}_0$: This is the hybrid game $i-1$. And then for all the indices $j \leq i-1$, $w_{j,\overline{M_j}} \stackrel{\$}{\leftarrow} \mathbb{G}$, while for the indices $j \geq i$, $w_{j,\overline{M_j}} \leftarrow (c^{r_{j,\overline{M_j}}} \cdot d^{s_{j,\overline{M_j}}}) \cdot (c^{r_{j,\overline{M_j}}} \cdot d^{s_{j,\overline{M_j}}})^{\xi}$.

Game $\mathbf{G}_1$: In this game, we compute $w_{i,\overline{M_i}}$ as:

$$w_{i,\overline{M}_i} = e_{i,\overline{M}_i}^{\alpha+\xi\alpha'} \cdot u_{i,\overline{M}_i}^{\beta+\xi\beta'} \cdot v_{i,\overline{M}_i}^{\gamma+\xi\gamma'}$$

instead of

$$w_{i,\overline{M}_i} = (c^{r_{i,\overline{M_i}}} \cdot d^{s_{i,\overline{M_i}}}) \cdot (c'^{r_{i,\overline{M_i}}} \cdot d'^{s_{i,\overline{M_i}}})^{\xi}.$$

This modification is purely syntactical, and this game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_2$: In this game, we pick $v_{i,\overline{M}_i}$ at random, instead of computing it as $v_{i,\overline{M}_i} = \hat{h}^{r_{i,\overline{M}_i}} h^{s_{i,\overline{M}_i}}$. This game is indistinguishable from the previous one, under the DDH assumption. Indeed, given a tuple $(g, \hat{h}, e_{i,\overline{M}_i}, v')$, we can set $v_{i,\overline{M}_i} = v' \cdot h^{s_{i,\overline{M}_i}}$; and if this tuple is a DDH tuple, $v_{i,\overline{M}_i}$ is computed as in the previous game, and otherwise, it is computed as in this game. Notice that the discrete logarithm $r_{i,\overline{M}_i}$ of $e_{i,\overline{M}_i}$ is not used, which enables to use the DDH assumption.

Game $\mathbf{G}_3$: In this game, we generate h as $h \leftarrow g^a$, $\hat{h} \leftarrow g^{\hat{a}}$, with $a, \hat{a} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. This modification is purely syntactical and this game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_4$: Then, for all commitments $C' = (e'_{j,b}, u'_{j,b}, v'_{j,b}, w'_{j,b})_{j,b}$ with label ℓ' queried to the oracle $\mathsf{C.Ext}$ or $\mathsf{C.IsBinding}$, each time we need to perform a test of the form:

$$w_{j',b'} \stackrel{?}{=} e'^{\alpha+\xi'\alpha'}_{j',b'} \cdot u'^{\beta+\xi'\beta'}_{j',b'} \cdot v'^{\gamma+\xi\gamma'}_{j',b'}, \tag{2}$$

where $\xi' = \mathcal{H}(\ell', (e'_{j,b}, u'_{j,b}, v'_{j,b})_{j,b})$, we reject the test as soon as: $v'_{j',b'} \neq u'^{a}_{j',b'} \cdot e'^{\hat{a}}_{j',b'}$.

This game is statistically indistinguishable from the previous one. To prove it, we use a sequence of hybrid games to add one by one this new test: for each commitment (in order of the queries to $\mathsf{C.Ext}$ and $\mathsf{C.IsBinding}$) and then each pair (j', b') (e.g., in lexicographical order). We remark that the newly added test for some C', j' and b', the only information (from an information theory point of view) the adversary has about $\alpha, \alpha', \beta, \beta', \gamma, \gamma'$ is at most:

- $\log c = \alpha + \gamma\hat{a}$ and $\log c' = \alpha' + \gamma'\hat{a}$ from the definition of c and c' (where log is the discrete logarithm in base g)
- $\log d = \beta + \gamma a$ and $\log d' = \beta' + \gamma' a$ from the definition of d and d'
- $\log w_{i,\overline{M}_i} = (\alpha + \xi\alpha') \log e_{i,\overline{M}_i} + (\beta + \xi\beta') \log u_{i,\overline{M}_i} + (\gamma + \xi\gamma') \log v_{i,\overline{M}_i}$, from the value of $w_{i,\overline{M}_i}$
- and values of the form $e''^{\alpha+\xi''\alpha'}_{j,b} \cdot u''^{\beta+\xi'\beta'}_{j,b} \cdot v''^{\gamma+\xi\gamma'}_{j,b}$ for commitments $C'' = (e''_{j,b}, u''_{j,b}, v''_{j,b}, w''_{j,b})_{j,b}$ with label ℓ'', queried before C'. But in this case, necessarily, $v''_{j,b} = u''^{a}_{j,b} \cdot e''^{\hat{a}}_{j,b}$, since otherwise this value would not have been computed. And so, this value can be computed by linear combinations of the previous equations (which can be seen easily on the matrix below).

Finally, we get that, from the adversary point of view, $\alpha, \alpha', \beta, \beta', \gamma, \gamma'$ are random values verifying the following system of equations:

$$\begin{pmatrix} \log c \\ \log c' \\ \log d \\ \log d' \\ \log w_{i,\overline{M}_i} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \hat{a} & 0 \\ 0 & 1 & 0 & 0 & 0 & \hat{a} \\ 0 & 0 & 1 & 0 & a & 0 \\ 0 & 0 & 0 & 1 & 0 & a \\ \log e_{i,\overline{M}_i} & \xi \log e_{i,\overline{M}_i} & \log u_{i,\overline{M}_i} & \xi \log u_{i,\overline{M}_i} & \log v_{i,\overline{M}_i} & \xi \log v_{i,\overline{M}_i} \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \alpha' \\ \beta \\ \beta' \\ \gamma \\ \gamma' \end{pmatrix}.$$

Since $\xi' \neq \xi$, if $v'_{j',b'} \neq u'^{a}_{j',b'} \cdot e'^{\hat{a}}_{j',b'}$, then

$$\begin{pmatrix} \log e'_{j',b'} & \xi' \log e'_{j',b'} & \log u'_{j',b'} & \xi' \log u'_{j',b'} & \log v'_{j',b'} & \xi' \log v'_{j',b'} \end{pmatrix}$$

is linearly independent of the rows of the above rectangular matrix, and so $e'^{\alpha+\xi'\alpha'}_{j',b'} \cdot u'^{\beta+\xi'\beta'}_{j',b'} \cdot v'^{\gamma+\xi\gamma'}_{j',b'}$ is completely random, from the adversary point of view. Therefore, with probability $1/p$, $w_{j',b'} \neq e'^{\alpha+\xi'\alpha'}_{j',b'} \cdot u'^{\beta+\xi'\beta'}_{j',b'} \cdot v'^{\gamma+\xi\gamma'}_{j',b'}$, and the test in Equation Equation (2) would also have failed. And adding this new test is statistically indistinguishable.

Game $\mathbf{G}_5$: In this game, we remark that, from the adversary point of view, before $w_{i,\overline{M}_i}$ is computed, $\alpha, \alpha', \beta, \beta', \gamma, \gamma'$ are random values verifying the following system of equations:

$$\begin{pmatrix} \log c \\ \log c' \\ \log d \\ \log d' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \hat{a} & 0 \\ 0 & 1 & 0 & 0 & 0 & \hat{a} \\ 0 & 0 & 1 & 0 & a & 0 \\ 0 & 0 & 0 & 1 & 0 & a \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \alpha' \\ \beta \\ \beta' \\ \gamma \\ \gamma' \end{pmatrix}.$$

Since, with high probability, $v_{i,\overline{M}_i} \neq u'^{a}_{i,\overline{M}_i} \cdot e'^{\hat{a}}_{i,\overline{M}_i}$ ($v_{i,\overline{M}_i}$ being chosen at random), then

$$\begin{pmatrix} \log e_{i,\overline{M}_i} & \xi' \log e'_{i,\overline{M}_i} & \log u'_{i,\overline{M}_i} & \xi' \log u'_{i,\overline{M}_i} & \log v'_{i,\overline{M}_i} & \xi' \log v'_{i,\overline{M}_i} \end{pmatrix}$$

is linearly independent of the rows of the above rectangular matrix, and so $w_{i,\overline{M}_i} = e'^{\alpha+\xi'\alpha'}_{i,\overline{M}_i} \cdot u'^{\beta+\xi'\beta'}_{i,\overline{M}_i} \cdot v'^{\gamma+\xi\gamma'}_{i,\overline{M}_i}$ is completely random, from the adversary point of view.

So, in this game, we replace $w_{i,\overline{M}_i}$ by a random value in $\mathbb{G}$, and this is statistically indistinguishable from the previous game.

Game $\mathbf{G}_6$: In this game, we now remove the extra tests introduced in $\mathbf{G}_4$. This game is statistically indistinguishable from the previous one, using a proof similar to the one in $\mathbf{G}_4$ (except this time, it is even easier, since $w_{i,\overline{M}_i}$ gives no information on $\alpha, \alpha', \beta, \beta', \gamma, \gamma'$ to the adversary.

Game $\mathbf{G}_7$: In this game, we compute again $v_{i,\overline{M}_i}$ as $v_{i,\overline{M}_i} = \hat{h}^{r_{i,\overline{M}_i}} h^{s_{i,\overline{M}_i}}$, instead of picking it at random. This game is computationally indistinguishable from the previous one under the DDH. The proof is similar to the one for $\mathbf{G}_1$.

Finally, we remark that this game is exactly the hybrid game i.

As a consequence, each hybrid step just involves the DDH assumption.

C.4 Strong Simulation Indistinguishability

The strong simulation indistinguishability can be proven using the following sequence of games:

Game $\mathbf{G}_0$: This is the game $\mathsf{Exp}^{\texttt{s-sim-ind-1}}_{\mathcal{A}}(\mathfrak{K})$ for strong simulation indistinguishability (for $b = 1$) recalled in Figure 3.

Game $\mathbf{G}_1$: In this game, for all queries $\mathsf{C.SCom}^\ell(\tau, \boldsymbol{M})$, we pick the values $w_{i,\overline{M}_i}$ at random (for all $i = 1, \ldots, m$). With an hybrid proof, applying the w-pseudo-randomness to all simulated commitments, this game is indistinguishable from the previous one.

Game $\mathbf{G}_2$: In this game, for all queries $\mathsf{C.SCom}^\ell(\tau, \boldsymbol{M})$, we pick $v_{i,\overline{M}_i}$ at random, instead of computing it as $v_{i,\overline{M}_i} = \hat{h}^{r_{i,\overline{M}_i}} h^{s_{i,\overline{M}_i}}$. This game is indistinguishable from the previous one, under the DDH assumption. Indeed, given a tuple $(g, \hat{h}, e_{i,\overline{M}_i}, v')$, we can set $v_{i,\overline{M}_i} = v' \cdot h^{s_{i,\overline{M}_i}}$; and if this tuple is a DDH tuple, $v_{i,\overline{M}_i}$ is computed as in the previous game, and otherwise, it is computed as in this game. Notice that the discrete logarithm $r_{i,\overline{M}_i}$ of $e_{i,\overline{M}_i}$ is not used, which enables to use the DDH assumption.

This last game is actually exactly the game $\mathsf{Exp}_{\mathcal{A}}^{\texttt{s-sim-ind-0}}(\mathfrak{K})$ for strong simulation indistinguishability (for $b = 0$) recalled in Figure 3.

C.5 Robustness

The robustness can be proven using the following sequence of games:

Game $\mathbf{G}_0$: This is the game $\mathsf{Exp}_{\mathcal{A}}^{\texttt{robust}}(\mathfrak{K})$ for robustness recalled in Figure 3.

Game $\mathbf{G}_1$: In this game, we answer all queries $\mathsf{C.SCom}^\ell(\tau, \boldsymbol{M})$ by $\mathsf{C.Com}^\ell(\boldsymbol{M})$. In other words, we replace all simulated commitments by normal ones. This game is indistinguishable from the previous one thanks to the strong simulation indistinguishability.

Game $\mathbf{G}_2$: In this game, we generate h as $h \leftarrow g^a$ and $\hat{h} \leftarrow g^{\hat{a}}$, with $a, \hat{a} \xleftarrow{\$} \mathbb{Z}_p$. This modification is purely syntactical and this game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_3$: In this game, we remark that if $\mathsf{C.IsBinding}^\ell(\tau, C, \boldsymbol{M})$ returns 0, and if $\boldsymbol{M} \xleftarrow{\$} \mathsf{C.Ext}^\ell(\tau, C)$, then there exists $i = i^*$ such that:

$$w_{i^*,b} = e_{i^*,b}^{\alpha+\xi\alpha'} \cdot u_{i^*,b}^{\beta+\xi\beta'} \cdot v_{i^*,b}^{\gamma+\xi\gamma'} \qquad \text{for } b = 0, 1.$$

And so, we abort the game if $v_{i^*,b} \neq u_{i^*,b}^{a} \cdot e_{i^*,b}^{\hat{a}}$ for $b = 0$ or $b = 1$. This game is statistically indistinguishable from the previous one. The proof is similar to the one for $\mathbf{G}_4$ in the proof for w-pseudo-randomness in Section C.3.

In this last game, we finally remark that $v_{i^*,0} \cdot v_{i^*,1} / (e_{i^*,0}^{\hat{a}} \cdot e_{i^*,1}^{\hat{a}}) = u_{i^*,0}^{a} \cdot u_{i^*,1}^{a} = h^t$. So if an adversary breaks this last game, we can break the CDH for the tuple (g, h, T), by not doing this last check $v_{i^*,b} \neq u_{i^*,b}^{a} \cdot e_{i^*,b}^{\hat{a}}$ (and so not knowing the discrete logarithm a of h) and simply returning $v_{i^*,0} \cdot v_{i^*,1} / (e_{i^*,0}^{\hat{a}} \cdot e_{i^*,0}^{\hat{a}})$ as a candidate CDH value (recall that the discrete logarithm t of T is no more used, while the discrete logarithm a of h is just used to abort the game).

C.6 Strong Pseudo-Randomness

To prove the strong pseudo-randomness, we use the following sequence of games:

Game $\mathbf{G}_0$: This game is the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-0}}$.

Game $\mathbf{G}_1$: In this game, before computing H', we compute $\boldsymbol{M'} \leftarrow \mathsf{C.Ext}^{\ell'}(\tau, C')$ and we abort if $\mathsf{C.IsBinding}^{\ell'}(\tau, C', \boldsymbol{M'}) = 0$.

This game is indistinguishable from the previous one thanks to the robustness.

Game $\mathbf{G}_2$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H' by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, the fact that if $\boldsymbol{M'} \neq \boldsymbol{M}$ and $\mathsf{C.IsBinding}^{\ell'}(\tau, C', \boldsymbol{M'}) = 1$, then $(\ell', C') \notin \mathcal{L}_{(\mathsf{crs}, \boldsymbol{M})}$, and the fact that H could have been computed as follows: $\delta \leftarrow \mathsf{C.Open}^\ell(\mathsf{eqk}, C, \boldsymbol{M})$ and $H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{crs}, \boldsymbol{M}), (\ell, C), \delta)$.

Game $\mathbf{G}_3$: In this game, we replace $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{C.Sim}^\ell(\tau)$ by $C \xleftarrow{\$} \mathsf{C.Com}^\ell(\boldsymbol{M''})$ for some arbitrary $\boldsymbol{M''} \neq \boldsymbol{M}$. This game is indistinguishable thanks to strong simulation indistinguishability (since eqk is not used, $\mathsf{C.Sim}$ could have been replaced by $\mathsf{C.SCom}$ with a $\boldsymbol{M''}$ as message).

Game $\mathbf{G}_4$: In this game, when $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H by a random value.
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that $\boldsymbol{M''} \neq \boldsymbol{M}$, $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M''}) = 1$ (since C is a real commitment to $\boldsymbol{M''}$) and so, that $(\ell, C) \notin \mathcal{L}_{(\mathsf{crs},\boldsymbol{M})}$.
Notice that we could not have done this if $\boldsymbol{M'} = \boldsymbol{M}$, since, in this case, we still need to use hk to compute the hash value H' of C'. We are handling this (tricky) case in the following games.

Game $\mathbf{G}_5$: Let $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b}$. In this game, we compute $v_{i,\overline{M''_i}}$ as $v_{i,\overline{M''_i}} = \hat{h}^{r_{i,\overline{M''_i}}} h^{s_{i,\overline{M''_i}}}$ instead of picking it at random in $\mathbb{G}$, for all i. This game is indistinguishable from the previous one, under the DDH assumption. Indeed, given a tuple $(g, \hat{h}, e_{i,\overline{M''_i}}, v')$, we can set $v_{i,\overline{M''_i}} = v' \cdot h^{s_{i,\overline{M''_i}}}$; and if this tuple is a DDH tuple, $v_{i,\overline{M''_i}}$ is computed as in this game, and otherwise, it is computed as in the previous game. Notice that the discrete logarithm $r_{i,\overline{M''_i}}$ of $e_{i,\overline{M''_i}}$ is not used, which enables to use the DDH assumption.

Game $\mathbf{G}_6$: In this game, we replace H by a random value, in the case $\boldsymbol{M'} = \boldsymbol{M}$. So now H will be completely random, in all cases (since it was already the case when $\boldsymbol{M'} \neq \boldsymbol{M}$).
Let $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b}$ and $C' = (e'_{i,b}, u'_{i,b}, v'_{i,b}, w'_{i,b})_{i,b}$. And let $\xi = \mathcal{H}(\ell, (e_{i,b}, u_{i,b}, v_{i,b})_{i,b})$ and $\xi' = \mathcal{H}(\ell', (e'_{i,b}, u'_{i,b}, v'_{i,b})_{i,b})$. Finally, we write $r'_{i,b} = \log e'_{i,b}$ and $s'_{i,b} = \log u'_{i,b}$ for all i, b, log being the discrete logarithm in base g. There are two cases:

1. for all i, $v'_{i,M_i} = \hat{h}^{r'_{i,M_i}} \cdot h^{s'_{i,M_i}}$. In this case, since C' extracts to $\boldsymbol{M}$, this means that
$$w'_{i,M_i} = (e'^{\alpha+\xi'\alpha'}_{i,M_i} \cdot u'^{\beta+\xi'\beta'}_{i,M_i} \cdot v'^{\gamma+\xi'\gamma'}_{i,M_i}),$$
and so from the definition of c and d, we have that:
$$w'_{i,M_i} = (c^{r'_{i,M_i}} \cdot d^{s'_{i,M_i}}) \cdot (c'^{r'_{i,M_i}} \cdot d'^{s'_{i,M_i}})^{\xi}.$$
This means that $(\ell', C') \in \mathcal{L}_{(\mathsf{crs},\boldsymbol{M})}$, and its hash value H' could be computed knowing only hp, $(r'_{i,M_i})_i$ and $(s'_{i,M_i})_i$. Therefore, the hash value H of C looks random by smoothness.
2. for some i, $v'_{i,M_i} \neq \hat{h}^{r'_{i,M_i}} \cdot h^{s'_{i,M_i}}$. Then since $v_{i,M_i} = \hat{h}^{r_{i,M_i}} \cdot h^{s_{i,M_i}}$, for the KV-SPHF (in Section C.2) the rows of the matrix Γ in Equation 1 (page 24) and the two following vectors
$$\begin{gathered}(e_{i,M_i}, e^{\xi}_{i,M_i}, u_{i,M_i}, u^{\xi}_{i,M_i}, v_{i,M_i}, w_{i,M_i})\\(e'_{i,M_i}, e'^{\xi'}_{i,M_i}, u'_{i,M_i}, u'^{\xi'}_{i,M_i}, v'_{i,M_i}, w'_{i,M_i})\end{gathered}$$
are independent. Then, even given access to the hash value H' of C' and the projection key hp, the hash value H of C looks perfectly random.

The following games are just undoing the modifications we have done, but keeping H picked at random

Game $\mathbf{G}_7$: Let $C = (e_{i,b}, u_{i,b}, v_{i,b}, w_{i,b})_{i,b}$. In this game, we pick $v_{i,\overline{M''_i}}$ at random, instead of computing it as $v_{i,\overline{M''_i}} = \hat{h}^{r_{i,\overline{M''_i}}} h^{s_{i,\overline{M''_i}}}$, for all i. This game is indistinguishable from the previous one, under the DDH assumption. Indeed, given a tuple $(g, \hat{h}, e_{i,\overline{M''_i}}, v')$, we can set $v_{i,\overline{M''_i}} = v' \cdot h^{s_{i,\overline{M''_i}}}$; and if this tuple is a DDH tuple, $v_{i,\overline{M''_i}}$ is computed as in the previous game, and otherwise, it is computed as in this game. Notice that the discrete logarithm $r_{i,\overline{M''_i}}$ of $e_{i,\overline{M''_i}}$ is not used, which enables to use the DDH assumption.

Game $\mathbf{G}_8$: In this game, we now compute C as originally using $\mathsf{C.Sim}$. This game is indistinguishable thanks to strong simulation indistinguishability.

Game $\mathbf{G}_9$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we compute H' as originally (as the hash value of C').
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that if $\boldsymbol{M'} \neq \boldsymbol{M}$ and $\mathsf{C.IsBinding}^{\ell'}(\tau, C', \boldsymbol{M'}) = 1$, then $(\ell', C') \notin \mathcal{L}_{(\mathsf{crs},\boldsymbol{M})}$.

Game $\mathbf{G}_{10}$: In this game, we do not extract $\boldsymbol{M'}$ from C' nor abort when $\mathsf{C.IsBinding}^{\ell}(\tau, C, \boldsymbol{M'}) = 0$.
Thanks to the robustness, this game is indistinguishable from the previous one.
We remark that this game is exactly the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-1}}$.

C.7 Details on the Comparison in Section 5.2

For the Canetti-Fischlin commitment scheme [CF01], we use a Pedersen commitment as a chameleon hash and multi-Cramer-Shoup ciphertexts to commit to multiple bits in a non-malleable way (see [ABB+13] for a description of the multi-Cramer-Shoup encryption scheme). We do not know a SPHF on such commitment, since the opening information of a Pedersen commitment is a scalar.

For the complexity of [ACP09], we consider a slight variant without one-time signtaure but using labels and multi-Cramer-Shoup ciphertexts, as in the scheme in [ABB+13]. The size of the projection key is computed using the most efficient methods in [ABB+13].

Commitments in [CF01, ACP09, ABB+13] were not described as without erasures, but slight variants of them are, as explained in Section 2.4.

Finally, we always suppose there exists a family of efficient collision-resistant hash functions (for efficiency reason, since DDH implies the existence of such families).

D Adaptive OT and PAKE

In this appendix, we describe our OT and PAKE protocols, with the complete security proofs: security holds in the UC-framework, against adaptive adversaries, without requiring reliable erasures. They make use of non-committing encryption.

D.1 Non-Committing Encryption Scheme

A ν_{NCE}-bit non-committing encryption scheme is defined by six algorithms:
- $\mathsf{NCE.Setup}(1^{\mathfrak{K}})$ generates the parameters $\mathsf{NCE.param}$ for the scheme, which taken as argument of the other algorithms (often implicitly);
- $\mathsf{NCE.KG}(\mathsf{NCE.param})$ generates an encryption key ek together with a decryption key dk;
- $\mathsf{NCE.Enc}(\mathsf{ek}, R)$ encrypts the plaintext $R \in \{0,1\}^{\nu_{\mathsf{NCE}}}$ into the ciphertext χ;
- $\mathsf{NCE.Dec}(\mathsf{dk}, \chi)$ decrypts the ciphertext χ, and output the corresponding plaintext R;
- $\mathsf{NCE.Sim}(\mathsf{NCE.param})$ generates an encryption key ek, a ciphertext χ together with an equivocation key $\mathsf{eqk}_{\mathsf{NCE}}$;
- $\mathsf{NCE.Open}(\mathsf{eqk}_{\mathsf{NCE}}, \mathsf{ek}, \chi, R)$ generates random coins r_{KG} for $\mathsf{NCE.KG}$ and r_{Enc} for $\mathsf{NCE.Enc}$ corresponding to R.

It has to verify the following properties:
- *Correctness.* For any parameter $\mathsf{NCE.param} \stackrel{\$}{\leftarrow} \mathsf{NCE.Setup}(1^{\mathfrak{K}})$, any honestly generated key pair $(\mathsf{ek}, \mathsf{dk}) \stackrel{\$}{\leftarrow} \mathsf{NCE.KG}(\mathsf{NCE.param})$, and any plaintext $R \in \{0,1\}^{\nu_{\mathsf{NCE}}}$, we have that: $\mathsf{NCE.Dec}(\mathsf{dk}, \mathsf{NCE.Enc}(\mathsf{ek}, R)) = R$ with overwhelming probability;
- *Simulation indistinguishability.* One cannot distinguish real keys $(\mathsf{ek}, \mathsf{dk})$ and ciphertexts χ (using $\mathsf{NCE.KG}$ and $\mathsf{NCE.Enc}$) from simulated ones (using $\mathsf{NCE.Sim}$ and $\mathsf{NCE.Open}$) even with access to the associated random coins. A scheme is said (t, ε)-simulation-indistinguishable if $\mathsf{Adv}^{\texttt{nc-sim-ind}}(t) \leq \varepsilon$ (see the experiments $\mathsf{Exp}_{\mathcal{A}}^{\texttt{nc-sim-ind-}b}(\mathfrak{K})$ in Figure 8).

This definition is a straightforward extension of the definition in [CDMW09] to multiple bits messages. As in [CDMW09], the definition directly implies that the scheme is semantically secure.

A ν_{NCE}-bit non-committing encryption scheme can be constructed using any single-bit non committing encryption scheme (such as the one in [CDMW09]) ν_{NCE} times.

D.2 3-Round Oblivious Transfer

In this section, we present our 3-round OT, UC-secure against adaptive adversaries, without erasures.

Remark 2. Though the new protocol uses our new commitment scheme, it could alternatively use the commitment scheme in [ABB+13], by just replacing $w_{i,b}$ by the last part of the Cramer-Shoup ciphertexts in these schemes. The proof would be very similar. This replacement may yield a more efficient scheme (under SXDH however) when ν_m is large, since the projection key in [ABB+13] is shorter than for our scheme and multiple projection keys need to be sent due to the generic transformation of SPHF to EPH.

$\mathsf{Exp}_{\mathcal{A}}^{\texttt{nc-sim-ind-}b}(\mathfrak{K})$
  $\mathsf{NCE.param} \xleftarrow{\$} \mathsf{NCE.Setup}(1^{\mathfrak{K}})$
  $(R, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{NCE.param})$
  if $b = 0$ **then**
    $r_{\mathsf{KG}}, r_{\mathsf{Enc}} \xleftarrow{\$}$
    $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{NCE.KG}(\mathsf{NCE.param}; r_{\mathsf{KG}})$
    $\chi \leftarrow \mathsf{NCE.Enc}(\mathsf{ek}, R; r_{\mathsf{Enc}})$
  else
    $(\mathsf{ek}, \chi, \mathsf{eqk}_{\mathsf{NCE}}) \xleftarrow{\$} \mathsf{NCE.Sim}(\mathsf{NCE.param})$
    $(r_{\mathsf{KG}}, r_{\mathsf{Enc}}) \xleftarrow{\$} \mathsf{NCE.Open}(\mathsf{eqk}_{\mathsf{NCE}}, \mathsf{ek}, \chi, R)$
  return $\mathcal{A}(\mathsf{st}, \mathsf{ek}, \mathsf{dk}, \chi, r_{\mathsf{KG}}, r_{\mathsf{Enc}})$

Fig. 8. Simulation Indistinguishability

Construction. The scheme is depicted in Figure 9. Our 1-out-of-k OT protocol uses a NCE channel of $\nu_{\mathsf{NCE}} = 2\nu_k + k\nu_m$ bits, where $k = 2^{\nu_k}$, for ν_m-bit strings . This channel is used to send a random value R. The last $k\nu_m$ bits of R are k ν_m-bit values $R_1, \ldots, R_k$. These values are used to mask the messages $m_1, \ldots, m_k$ sent by the sender, to be able to reveal the correct messages, in case of corruption of the sender (when both the sender and the receiver were honest at the beginning, and so when $m_1, \ldots, m_k$ were completely unknown to the simulator).

The first $2\nu_k$ bits of R are used to make the commitment C (which is normally simulated when the receiver is honest) perfectly binding to the revealed index s, in case of corruption of the receiver (when both the sender and the receiver were honest at the beginning, and so when s was completely unknown to the simulator). More precisely, they are used to partially hide the last component of commitments: the $w_{i,b}$; the bit R_{2i+b-1} indicates whether $w_{i,b}$ has to be inverted or not before use.

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{C.Setup}(1^{\mathfrak{K}})$ and $\mathsf{NCE.param} \xleftarrow{\$} \mathsf{NCE.Setup}(1^{\mathfrak{K}})$.

Pre-flow:
1. P_i generates $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{NCE.KG}(\mathsf{NCE.param})$
2. P_i sends ek to P_j

Index query on s**:**
1. P_j chooses a random $R \xleftarrow{\$} \{0,1\}^{\nu_{\mathsf{NCE}}}$ and computes $\chi \xleftarrow{\$} \mathsf{NCE.Enc}(\mathsf{ek}, R)$
2. P_j computes $(C = ((e_{I,b}, u_{I,b}, v_{I,b}, w_{I,b})_{I,b}), \delta) \xleftarrow{\$} \mathsf{C.Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. P_j sets $w'_{I,b} = w_{I,b}$ if $R_{2I+b-1} = 0$ and $w'_{I,b} = 1/w_{I,b}$ otherwise, for $I = 1, \ldots, \nu_k$ and $b = 0, 1$; and sets $C' = ((e_{I,b}, u_{I,b}, v_{I,b}, w'_{I,b})_{I,b})$
4. P_j sends χ and C' to P_i

Database input $(m_1, \ldots, m_k)$**:**
1. P_i computes $R \xleftarrow{\$} \mathsf{NCE.Dec}(\mathsf{dk}, \chi)$
2. P_i sets $w_{I,b} = w'_{I,b}$ if $R_{2I+b-1} = 0$ and $w_{I,b} = 1/w'_{I,b}$ otherwise, for $I = 1, \ldots, \nu_k$ and $b = 0, 1$; and sets $C = ((e_{I,b}, u_{I,b}, v_{I,b}, w_{I,b})_{I,b})$
3. P_i sets $(R_t)_t$ to the last $k\nu_m$ bits of R (R_t being a ν_m-bit variable)
4. P_i computes $\mathsf{hk}_t \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs})$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, \mathsf{crs}, (\ell, C))$, $K_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, (\mathsf{crs}, t), (\ell, C))$, and $M_t \leftarrow R_t \operatorname{xor} K_t \operatorname{xor} m_t$, for $t = 1, \ldots, k$
5. P_i sends $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$

Data recovery:
Upon receiving $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$, P_j computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (\mathsf{crs}, s), (\ell, C), \delta)$ and gets $m_s \leftarrow R_s \operatorname{xor} K_s \operatorname{xor} M_s$, with $(R_t)_t$ the last $k\nu_m$ bits of R.

Fig. 9. UC-Secure 1-out-of-k OT from our SPHF-Friendly Commitment for Adaptive Adversaries

Security Proof. The proof is similar to the semi-adaptive one in Section B.1, except with games dealing with honest receivers talking to honest senders. More precisely, the game $\mathbf{G}_5$ is no more indistinguishable from the previous one, and we replace it with the following sequence of games:

Game $\mathbf{G}_0$**:** When simulating an **honest sender** P_i, instead of honestly computing the keys $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{NCE.KG}(\mathsf{NCE.param})$, we use the simulation: $(\mathsf{ek}, \chi, \mathsf{eqk}_{\mathsf{NCE}}) \xleftarrow{\$} \mathsf{NCE.Sim}(\mathsf{NCE.param})$, and $(r_{\mathsf{KG}}, r_{\mathsf{Enc}}) \xleftarrow{\$} \mathsf{NCE.Open}(\mathsf{eqk}_{\mathsf{NCE}}, \mathsf{ek}, \chi, R)$ with some random $R \xleftarrow{\$} \{0,1\}^{\nu_{\mathsf{NCE}}}$. Finally, we set

$(\mathsf{ek}, \mathsf{dk})$ to $\mathsf{NCE.KG}(\mathsf{NCE.param}; r_{\mathsf{KG}})$, which should not change ek, otherwise the non-committing encryption scheme would not be simulation indistinguishable. Then, if P_j is not corrupted when he received the pre-flow from P_i, we use the previously computed χ instead of computing a new one.
This game is indistinguishable from the previous one thanks to simulation indistinguishability.
We remark that the computation of dk using r_{KG} is actually only used when P_i receives a flow from a corrupted receiver, in which case he needs to be able to decrypt the ciphertext χ sent by the adversary. In all cases, dk and r_{KG} only need to be computed in case of corruption of P_i and P_j, and so R may be modified depending on the inputs learned by the corruption (inputs which are already known in this game but will not be known at in the last game). Intuitively, the only restriction is to ensure that R looks random to the adversary.

Game $\mathbf{G}_1$: We still deal with an **honest sender** P_i. If P_i receives an honest flow from P_j, we now pick M_t at random, and then set $R_t = M_t \operatorname{xor} K_t \operatorname{xor} m_t$ (for all t). Recall that R_t is part of R and only needs to be revealed in case of corruption of P_i and P_j.
This game is perfectly indistinguishable from the previous one.
We remark that, in this game, as long as P_i and P_j remains uncorrupted, all flows seen by the adversary are completely independent of the messages m_t and the hash values K_t.

Game $\mathbf{G}_2$: We now deal with the case where an **honest receiver** P_j gets corrupted while its associated sender P_i **is still honest**. If the corruption is before P_i sent his flow, there is nothing to do. Let us focus on the case where the corruption is after P_i sent his flow.
In this case, we learn the index query s (we already knew). We write $s = \boldsymbol{M}$, and we flip the bits $R_{2I+\overline{M_I}+1}$, in such a way this makes the resulting commitment binding to $\boldsymbol{M}$. Indeed, $w_{I,b} = w'_{I,b}$ if $R_{2I+b-1} = 0$ and $w_{I,b} = 1/w'_{I,b}$ otherwise; and so flipping this bits make $w_{I,\overline{M_I}}$ invalid, while w_{I,M_I} stays valid.
Then, we can just compute the state accordingly to this commitment, i.e., compute K_t as the hash value of this commitment, then compute $R_t = M_t \operatorname{xor} K_t \operatorname{xor} m_t$, and finally we can set $(r_{\mathsf{KG}}, r_{\mathsf{Enc}}) \stackrel{\$}{\leftarrow} \mathsf{NCE.Open}(\mathsf{eqk}_{\mathsf{NCE}}, \mathsf{ek}, \chi, R)$. We recall that R_t is a part of R, and that P_i being still honest, we know all m_t.
This game is indistinguishable from the previous one, thanks to the w-pseudo-randomness of our commitment scheme (see Figure 7).

We then remark that after this game, when two honest users stay honest for the whole time, the simulator does not need their inputs, since they are only required in case of corruption. In addition, when one user P gets corrupted in a sub-session where both users were initially honest, then the revealed internal state of P corresponds nearly to the one a real user following the protocol with the real inputs would get. More precisely, if we omit the fact that ek and χ are simulated, this is exactly the case for an honest sender P_i. For an honest P_j, there is only one difference: the values $v_{i,b}$ of the commitments are all "valid" (i.e., $v_{i,b} = \hat{h}^{\log e_{i,b}} \cdot h^{\log u_{i,b}}$) while for a real user $v_{i,\overline{M_i}}$ would be completely random. However, the resulting commitment is still perfectly binding, which is very important to be able to explain hk_t for $t \neq s$ in case of later corruption of the sender P_i.

So, the original sequence of games for semi-adaptive adversaries (from $\mathbf{G}_7$ will also work with minor modifications. Here are the modifications:

- in $\mathbf{G}_7$, we do not extract or call $\mathsf{C.IsBinding}$ on a commitment C generated by an honest receiver (even if the receiver is now corrupted).
- in $\mathbf{G}_8$, we only apply the modifications in this game when at least one player is corrupted. The modifications still makes this game indistinguishable from the previous one, even when C was generated by an honest receiver, since such commitments are also perfectly binding (as recalled above).

Table 2. Comparison of 1-out-of-k OT UC-Secure against Adaptive Adversaries, without Erasures, with $k = 2^{\nu_k}$

	Rounds	Communication Complexity
[GWZ09]	≥8	$(k+1)\cdot m \times \mathrm{NCE} + 3\cdot(2^k+2k)\cdot m \times \mathbb{G} + (2^k+2k)\cdot\big(\mathrm{com}(4\times\mathbb{G}) + 2\nu_k\times\mathbb{G} + \nu_k \times \mathrm{ZK} + 4m\nu_k\times\mathbb{G}\big)$
1st	4	$(k+1)\cdot m \times \mathrm{NCE} + 3\cdot(2^k+2k)\cdot m\times\mathbb{G} + (2^k+2k)\cdot\big(7\nu_k\times\mathbb{G} + m\cdot(2\times\mathbb{G} + (\mathbb{Z}_p)^{\mathrm{b}} + 2)\big)$
2nd	3	$(km + 2\nu_k)\times\mathrm{NCE} + 7\nu_k\times\mathbb{G} + m\cdot\big(2\times\mathbb{G} + (\mathbb{Z}_p)^{\mathrm{b}} + 2\big)$

[a] number of rounds
[b] this element in $\mathbb{Z}_p$ is not required when $\nu_m = \nu_k = 1$
Legend:
- ZK: zero-knowledge proof used in [GWZ09].
- com(x): communication complexity of a UC-commitment scheme for x bits. This is used to generate the CRS for the scheme in [PVW08]. If this commitment is interactive, this increases the number of required rounds.
- $x \times$ NCE: x bits sent by non-committing encryption scheme.

D.3 Comparison of Adaptive OT Schemes

In Table 2, we compare our OT schemes with the DDH-based OT in [GWZ09]. The QR-based one in less efficient anyway. A summary of this table can be found in Section 6.1.

We suppose we use the non-committing encryption (NCE) scheme proposed in [CDMW09] (which is 2-round) and the ElGamal encryption as simulation encryption scheme for the NCE scheme and the somewhat NCE construction (which also requires a simulation encryption scheme). So all our schemes are secure under DDH (plus existence of collision resistant hash functions and symmetric key encryption, but only for efficiency, since DDH implies that also).

In the comparison, we extend the schemes in [GWZ09] to 1-out-of-k schemes using the method explained in Section 6.1 and the 1-out-of-k version of the schemes of Peikert *et al.* [PVW08], which consists in doing ν_k schemes in parallel and secret sharing the messages (where $k = 2^{\nu_k}$).

To understand the costs in the table, recall that a 2^l-somewhat non-committing encryption scheme works as follows: one player sends a l-bit value I using a full NCE scheme (2 rounds) together with 2^l public keys all samples obviously except the I^{th} one, and then the other player sends 2^l ciphertexts samples obliviously except the I^{th} one which contains a symmetric key K. Then to send any message through this 2^l-somewhat NCE channel, a player just sends 8 messages all random except the I^{th} one which is an encryption of the actual message under K. This means that if the original semi-adaptive protocol is x-round, then the protocol resulting from the transformation of Garay *et al.*, is $(x+2)$-round; and this costs a total of $3\cdot 2^l$ group elements, in addition of the group elements for the l-bit non-committing encryption.

D.4 Password-Authenticated Key Exchange

In this section, we present two PAKE constructions UC-secure against adaptive adversaries: a (very) inefficient 3-round PAKE and an efficient TODO-round PAKE Remark 2 (page 30) also applies. Please notice that slightly more efficient variants can be constructed using more rounds, since if the projection keys can be sent after the commitments, only GL-EPHFs are needed and GL-EPHFs are much more efficient than KV-EPHFs. This remark also holds for our semi-adaptive PAKE.

Construction Optimized for Round Complexity. The scheme is depicted in Figure 10. It uses a NCE channel with ν_{NCE} bits, where ν_{NCE} is the number of bits exchanged by the two players in the semi-adaptive protocol (plus a one-time signature). The value R is divided in two parts R_1 and R_2: the first one is used to mask the first flow, while the second one is used to mask the second flow.

The security proof is straightforward from the semi-adaptivity of the underlying PAKE.

The scheme can be improved by replacing the KV-EPHF to hash C_j by a GL-EPHF, which is possible since P_i receives C_j before computing hp_j. Even with this modification, this protocol remains highly inefficient.

CRS: $\mathsf{crs} \stackrel{\$}{\leftarrow} \mathsf{C.Setup}(1^{\mathfrak{K}})$.

Pre-flow (by P_i):

1. P_i generates $(\mathsf{vk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{OT.KG}(1^{\mathfrak{K}})$
2. P_i generates $(\mathsf{ek}, \mathsf{dk}) \stackrel{\$}{\leftarrow} \mathsf{NCE.KG}(\mathsf{NCE.param})$
3. P_i sends vk and ek to P_j

First flow (by P_j with password π_j):

1. P_j chooses a random $R \stackrel{\$}{\leftarrow} \{0,1\}^{\nu_{\mathsf{NCE}}}$ and computes $\chi \stackrel{\$}{\leftarrow} \mathsf{NCE.Enc}(\mathsf{ek}, R)$
2. P_j sets R_1 to the first part of R and R_2 to the second part (see text)
3. P_j generates $\mathsf{hk}_j \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$, $\mathsf{hp}_j \leftarrow \mathsf{ProjKG}(\mathsf{hk}_j, \mathsf{crs}, \perp)$
4. P_j computes $C_j \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell_j}(\pi_j)$ with $\ell_j = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{vk}, \mathsf{ek}, \mathsf{hp}_j)$
5. P_j sends χ, $F_j = (\mathsf{hp}_j, C_j) \operatorname{xor} R_1$ to P_i

Second flow (by P_i with password π_i):

1. P_i computes $R \leftarrow \mathsf{NCE.Dec}(\mathsf{ek}, \chi)$
2. P_i sets R_1 to the first part of R and R_2 to the second part (see text)
3. P_i computes $(\mathsf{hp}_j, C_j) \leftarrow F_j \operatorname{xor} R_1$
4. P_i generates $\mathsf{hk}_i \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathsf{crs}, \perp)$
5. P_i computes $C_i \stackrel{\$}{\leftarrow} \mathsf{C.Com}^{\ell_i}(\pi_i)$ with $\ell_i = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{vk})$
6. P_i computes $\sigma \stackrel{\$}{\leftarrow} \mathsf{OT.Sign}(\mathsf{sk}, (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{ek}, \chi, \mathsf{hp}_j, C_j, \mathsf{hp}_i, C_i))$
7. P_i sends $F_i = (\mathsf{hp}_i, C_i, \sigma) \operatorname{xor} R_2$ to P_j

Key computation for P_i:

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathsf{crs}, \pi_i), (\ell_i, C_i), \delta_i)$
   and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathsf{crs}, \pi_i), (\ell_j, C_j))$ with $\ell_j = (\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \mathsf{hp}_j, \mathsf{vk}, \mathsf{ek}, \mathsf{hp}_j)$
2. P_i computes $\mathsf{SK}_i = H'_i \operatorname{xor} H_j$

Key computation for P_j:

1. P_j computes $(\mathsf{hp}_i, C_i, \sigma) \leftarrow F_i \operatorname{xor} R_2$
2. P_i checks that $\mathsf{OT.Verify}(\mathsf{vk}, \sigma, (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{ek}, \chi, \mathsf{hp}_j, C_j, \mathsf{hp}_i, C_i)) = 1$, and aborts if it is not the case
3. P_i performs a computation similar to the one of P_j to get SK_j.

Fig. 10. UC-Secure PAKE from an SPHF-Friendly Commitment for Adaptive Adversaries

Construction Optimized for Communication Complexity. This construction is actually generic and can transform any PAKE UC-secure against semi-adaptive adversaries into a UC-secure PAKE UC-secure against adaptive adversaries.

Basically it works as follows. First, the two players exchanged verification keys vk_i and vk_j for a signature scheme. Then, as in [BCL+05], each player signs his flows together with the previous flows, using his signature key. This provides a kind of weakly authenticated channel.

Then, the two players run m semi-adaptive PAKE, one for each bit $\pi[k]$ of their password, each PAKE being run inside a 4-somewhat non-committing encryption scheme. In addition, one player will send a $\mathfrak{K}$-bit random value R using a fully non-committing encryption scheme. The final shared key is the xor of all keys of all PAKE protocols and R.

If we ignore the problems of non-authenticated channels (solved using signatures), when $m = 1$ this protocol is an efficient variant of the Garay *et al.* [GWZ09] transformation, where we only use the somewhat non-committing encryption channels to deal with the inputs. Remark that the original transformation would have required a $2^{\mathfrak{K}+2}$-somewhat non-committing encryption scheme, which is impossible to realize.

The security proof is very similar to the one for the transformation of Garay *et al.* [GWZ09]. Basically, when both parties are hones, in each 4-somewhat non-committing encryption channel, the simulator puts 4 versions of the protocol: depending which party gets corrupted first and which was its password bit $\pi[k]$. In case of corruption, it reveals the correct version of the protocol and chooses R to match the revealed shared key.

E A Construction of EPHF with Hint and Application to OT

In this section, we propose an efficient GL-EPHF with hint from any SPHF constructed using the generic framework of [BBC+13b] in some cyclic group $\mathbb{G}$, as long as τ enables to compute the discrete logarithms (in some base g) of the matrix Γ (in the framework). We then use this new construction to propose a variant of our semi-adaptive OT.

In the sequel, we use the same notations as in [BBC+13b].

We suppose that we have a symmetric bilinear group $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, where e is a bilinear map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. Notice that we cannot directly use our SPHF-friendly commitment or the one in [ABB+13], since in such a group the DDH assumption does not hold. But we can easily adapt this SPHF-friendly commitments to symmetric groups.

E.1 Recall of the Generic Framework

In the generic framework of [BBC+13b], an SPHF is defined by a full-rank matrix $\Gamma \in \mathbb{G}_1^{k\times n}$ (which depends on crs for $\mathsf{KV\text{-}SPHF}$, and on crs and the word C for $\mathsf{GL\text{-}SPHF}$), a function $\Theta : \mathcal{X} \to \mathbb{G}^{1\times n}$ (which depends on $\mathsf{full\text{-}aux}$ and in case of $\mathsf{GL\text{-}SPHF}$, also possibly on an additionnal value ε which is a part of hp and hk[4]). We have the following property, for any aux and any $C \in \mathcal{X}$, with high probability over ε (if ε is used):

$$C \in \mathcal{L}_{\mathsf{full\text{-}aux}} \iff \exists \boldsymbol{\lambda} \in \mathbb{Z}_p^k,\ \Theta(C) = \boldsymbol{\lambda} \odot \Gamma,$$

where $\oplus$ and $\odot$ are the natural operations on the field $\mathbb{Z}_p$ and the groups $\mathbb{G}$ and $\mathbb{G}_T$ (details can be found in [BBC+13b]). In addition $\boldsymbol{\lambda}$ can efficiently be computed from the witness of C in $\mathcal{L}_{\mathsf{full\text{-}aux}}$.

The hashing key hk of the SPHF is a random vector $\boldsymbol{\alpha} \in Z_p^n$ (or $\mathsf{hk} = (\boldsymbol{\alpha}, \varepsilon)$ when $\neq$ is used), while the projection key hp is the vector $\Gamma \odot \boldsymbol{\alpha}$ (or $\mathsf{hp} = (\boldsymbol{\gamma}, \varepsilon)$ when $\neq$ is used). And the hash value is:

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \coloneqq \Theta(C) \odot \boldsymbol{\alpha} = \boldsymbol{\lambda} \odot \boldsymbol{\gamma} =: \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w).$$

In the sequel, we ignore ε for the sake of simplicity.

E.2 GL-EPHF with Hint

In this section, we describe our $\mathsf{GL\text{-}EPHF}$ with Hint.

Basically, we want to be able to choose $\mathsf{hp} = \boldsymbol{\gamma}$, given C, such that given any hash value H, we can find $\boldsymbol{\alpha}$ such that:

$$\begin{aligned} \Gamma \odot \boldsymbol{\alpha} &= \boldsymbol{\gamma} \\ \Theta(C) \odot \boldsymbol{\alpha} &= H. \end{aligned}$$

Recall that the second one is much harder, because C is generated by the adversary and we have no way to know its discrete logarithm.

To solve this issue, instead of choosing $\boldsymbol{\alpha}$ at random in $\mathbb{Z}_p^n$, we choose it in $\mathbb{G}^n$. Now, the operation $\odot$ above corresponds to the pairing operation, and H and $\boldsymbol{\gamma}$ are in $\mathbb{G}_T^k$.

Let us suppose that Γ is a full-rank matrix of $k = n - 1$ rows and n columns. If Γ has less than $n - 1$ rows, we can always add "fake" random rows to Γ, and $\Theta(C)$ will still be linearly independent of the rows of Γ.

Let us write $\hat{\Gamma}$ the matrix containing the discrete logarithm of the coefficients of Γ. Let M be the following square matrix: $M = \begin{pmatrix} \hat{\Gamma} \\ \Theta(C) \end{pmatrix}$. All its entries are in $\mathbb{Z}_p$ except in the last row.

Let $\tilde{M}$ be the comatrix of M and $\det M$ be the determinant of M. We remark that $\det M \in \mathbb{G}$ and the entries of $\tilde{M}$ are all in $\mathbb{G}$ except its last column. In addition $\tilde{M}$ and $\det M$ can be computed efficiently if we know $\hat{\Gamma}$. Finally, we have $M \odot \tilde{M} = (\det M) I$ where I is the identity matrix.

Then SimKG chooses a random vector $\hat{\boldsymbol{\gamma}} \in \mathbb{Z}_p^k$ and sets:

$$\mathsf{hp} = \boldsymbol{\gamma} \coloneqq \hat{\boldsymbol{\gamma}} \odot (\det M) \qquad H' \coloneqq \det M \qquad \mathsf{expk} \coloneqq (\hat{\boldsymbol{\gamma}}, H', \tilde{M}).$$

[4] This ε can be used to do efficient conjunctions of SPHF as in Section C.2. It is not present in the original framework, but can easily be added to it.

We can compute the hashing key $\mathsf{hk} = \boldsymbol{\alpha}$ for a value $H \in \mathbb{G}_T$ with discrete logarithm hint in base $H' = \det M$ (i.e., $H = H' \odot \mathsf{hint}$), as:

$$\boldsymbol{\alpha} = \tilde{M} \odot \begin{pmatrix} \hat{\gamma} \\ \mathsf{hint} \end{pmatrix} \in \mathbb{G}^n.$$

Indeed, we have:

$$M \odot \boldsymbol{\alpha} = H' \odot \begin{pmatrix} \hat{\gamma} \\ \mathsf{hint} \end{pmatrix} = \begin{pmatrix} \gamma \\ H \end{pmatrix}.$$

E.3 Application to OT

In OT, if we slightly change the protocol in Section 4.2, we use our previous construction of GL-EPHF with hint. Essentially, what we need is to be able to compute the discrete logarithm hint of all the hash values which need to explain.

Therefore, instead of computing $M_t = K_t \operatorname{xor} m_t$, we just compute: $M_t = K_t^{m_t}$, assuming messages m_t are small (for decryption) and non-zero. To decrypt, the receiver needs to find the discrete logarithm of M_t in base K_t. For that purpose, he can use the variant of the Pollard's kangaroo method in [MT09]. The decryption complexity is therefore $O(2^{\nu_m/2})$. As for the generic version, we can use longer messages (than the one which could be decrypted using Pollard's algorithm) by using multiple projection keys and multiple M_t and K_t.

For the security proof, we can generate M_t as H'^{r_t} with r_t random, and when we need to explain it for m_t, we remark that:

$$M_t = (H'^{r_t/m_t})^{m_t},$$

and so we can use the explainability property of the GL-EPHF with $H = H'^{r_t/m_t}$ and $\mathsf{hint} = r_t/m_t$.

Recall that the generic version in Section 4.2 requires the simulator to do a brute force of 2^ν operations, where ν is the size of hash value. Here, the receiver needs to do $O(2^{\nu/2})$ operations. This saves a factor 2 and the proof is now tighter in some way. But unfortunately, the fact that the DDH assumption does not hold and that we need to use DLin variants of our protocols makes this protocol only roughly as efficient as the original one.

Designing a more efficient GL-EPHF with hint based, e.g., on SXDH is an open problem.